Naval Research Laboratory

Stennis Space Center, MS 39529-5004

NRL/MR/7320--12-9407

User's Guide for the Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS) Version 5.0, Rev. 2.0

Travis A. Smith
Timothy J. Campbell
Richard A. Allard
Ocean Dynamics and Prediction Branch
Oceanography Division

Suzanne N. Carroll
QinetiQ North America
Stennis Space Center, Mississippi

May 3, 2012

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
03-05-2012	Memorandum Report	

4. TITLE AND SUBTITLE

User's Guide for the Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS) Version 5.0, Rev. 2.0

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N

6. AUTHOR(S)

Travis A. Smith, Timothy J. Campbell, Richard A. Allard, and Suzanne N. Carroll*

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
73-4320-02-5

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Oceanography Division
Stennis Space Center, MS 39529-5004

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7320--12-9407

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
875 North Randolph Street, Suite 1425
Arlington, VA 22203-1995

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*QinetiQ North America, Stennis Space Center, Mississippi

14. ABSTRACT

The Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS) Version 5.0 is a fully coupled, data assimilative model which consists of the atmospheric model (COAMPS), coupled to the Navy Coastal Ocean Model (NCOM), and the Simulating Waves Nearshore (SWAN) model. Coupling of the models is accomplished via the Earth System Modeling Framework (ESMF), in which surface fluxes of momentum and moisture, sea surface heights, currents, Stokes' drift currents, and wave radiation stress gradients are exchanged across the air-sea-wave interface. This version of the user's manual includes instructions to setup COAMPS Version 5.0 on a user's workstation or on the High Performance Computing Modernization Program (HPCMP) DoD Supercomputing Resource Center (DSRC) platforms. Step-by-step instructions include using subversion control to upload COAMPS Version 5.0 code and namelists, setup of COAMPS, NCOM, and SWAN input/output directories and namelists, input data acquisition, creation of high-resolution grid setup namelists, and execution of COAMPS Version 5.0.

15. SUBJECT TERMS

ESMF	COAMPS	SWAN
Coupled	NCOM	

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Travis A. Smith
Unclassified	Unclassified	Unclassified	Unclassified Unlimited	174	**19b. TELEPHONE NUMBER** *(include area code)* (228) 688-5631

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

TABLE OF FIGURES AND TABLES

1.0 INTRODUCTION

The Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS®[i]) Version 5 is a coupling of the COAMPS Version 3 model with the Navy Coastal Ocean Model (NCOM) Version 4.0. This is accomplished through the Earth System Modeling Framework (ESMF). In this fully coupled mode, COAMPS and NCOM models can be integrated concurrently so that precipitation and surface fluxes of moisture and momentum are exchanged across the air-sea interface.

COAMPS receives input from the Navy Operational Global Atmospheric Prediction System (NOGAPS) (Hogan and Rosmond, 1991) in the form of meteorological observations, satellite data, and ship reports. It acquires ocean observations and bathymetry from the Global Navy Coastal Ocean Model (NCOM). Atmospheric and oceanographic model output includes surface and upper air parameters, sea surface temperature (SST), 3D temperature, salinity, velocity, 2D mixed layer depth, and acoustic products. Coupling can be both one-way and two-way as well as with or without data assimilation. Data assimilation is facilitated through the Navy Coastal Ocean Data Assimilation (NCODA) system (Cummings, 2005).

1.1 Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS®)

The Naval Research Laboratory (NRL) Marine Meteorology Division's COAMPS® makes both mesoscale and microscale predictions of the atmosphere and ocean. The atmospheric elements of COAMPS are used operationally by the U.S. Navy for numerical weather prediction in various regions around the world. COAMPS consists of a complete 3D multivariate optimal interpolation (MVOI) data assimilation system with data quality control, analysis, initialization, and forecast model components (Hodur, 1997; Chen et al., 2003). It can also accommodate 3D variational analysis through NAVDAS, the NRL Atmospheric Variational Data Assimilation System. Data assimilation is initiated by the prior 12-hr forecast and incorporates quality-controlled observations from aircraft, radiosondes, satellite, ship, and surface stations. The MVOI analysis employs both in situ and satellite SST measurements (Pullen et al., 2007). COAMPS includes a globally relocatable grid, user-defined grid resolutions and dimensions, nested grids, an option for idealized or real-time simulations, and code that allows for portability between mainframes and workstations. The nonhydrostatic atmospheric model has predictive equations for momentum, non-dimensional pressure perturbation, turbulent kinetic energy, potential temperature, and the mixing ratios of water vapor, clouds, rain, snow, and ice. It includes Louis surface flux parameterizations for boundary layer processes, precipitation, and radiation (Louis et al., 1981). COAMPS has 30 vertical terrain following levels.

Manuscript approved March 26, 2012.

1.2 Navy Coastal Ocean Model (NCOM)

The NCOM is based primarily on two existing ocean circulation models: the Princeton Ocean Model (POM) (Blumberg and Mellor, 1983; Blumberg and Mellor, 1987) and the Sigma/*Z*-level Model (SZM) (Martin et al., 1998). NCOM Version 4.0 has a free-surface and is founded on the primitive equations and the hydrostatic, Boussinesq, and incompressible approximations. The Mellor Yamada Level 2 (MYL2) and Mellor Yamada Level 2.5 (MYL2.5) turbulence models parameterize the vertical mixing. The Large et al. (1994) vertical mixing enhancement scheme is available for parameterization of unresolved mixing processes occurring at near-critical Richardson numbers. A source term included in the model equations allows for the input of river and runoff inflows.

NCOM employs a staggered Arakawa C grid (as in POM). Spatial finite differences are mostly second-order centered, but higher-order spatial differences are optional. NCOM has a leapfrog temporal scheme with an Asselin filter to suppress timesplitting. Most terms are managed explicitly in time, but the propagation of surface waves and vertical diffusion are implicit.

NCOM 4.0 has an orthogonal-curvilinear horizontal grid and a hybrid sigma and *z*-level grid with sigma coordinates applied from the surface down to a specified depth. It has level coordinates below the specified depth. A second, newer, choice is the general vertical coordinate (GVC) grid consisting of a three-tiered vertical structure. The GVC grid is comprised of: (1) a "free" sigma grid near the surface that expands and contracts with the movement of the free surface, (2) a "fixed" sigma grid that is immobile within the free surface, and (3) a *z*-level grid that allows for "partial" bottom cells (Martin et al., 2008a,b). The relocatable version of NCOM, known as RELO NCOM, is used to generate grids and namelists of parameters for NCOM.

1.3 Wave Models

1.3.1 Simulating WAves Nearshore (SWAN) Model

The Simulating WAves Nearshore (SWAN) model is a third-generation, phase-averaged system based on the spectral action balance equation treated in discrete form. It can simulate coastal zones with shallow water, (barrier) islands, local winds, tidal flats, and ambient currents. Though applicable at any scale, it is most efficient when predicting wave conditions in small scale. Short-crested, random wave fields propagating simultaneously from vastly differing directions can be accommodated. The SWAN model accounts for shoaling and refractive propagation (both depth and current induced), bottom friction and depth-induced wave breaking, wave generation due to wind, energy dissipation due to whitecapping, and nonlinear wave-wave interactions. SWAN is stationary, optionally nonstationary, and formulated in Cartesian or spherical coordinates. The stationary mode is used only for waves with a relatively short residence time in the computational area. In other words, wave travel time through the region is small compared to the time scale of the

geophysical conditions such as wave boundary conditions, wind, tides, and storm surge. A quasi-stationary approach is used with stationary SWAN computations in a time-varying sequence of static conditions (SWAN Team, 2006). A validation study of SWAN was conducted in 2004 by Allard et al.

1.3.2 WAVEWATCH III (WW3) Model

WAVEWATCH III™ (WW3) (Tolman 2009) is a third generation wave model similar to WAve Modelling (WAM) (Komen et al., 1994). It was developed at the National Oceanic and Atmospheric Administration's (NOAA) National Centers for Environmental Prediction (NCEP). WAVEWATCH III evolved from the WAVEWATCH and WAVEWATCH II models, created at Delft University of Technology (Tolman 1989, 1991a) and National Aeronautics and Space Administration (NASA), Goddard Space Flight Center (e.g., Tolman 1992), respectively. WAVEWATCH III, however, employs different governing equations, physical parameterizations, model structure, and numerical methods. It solves the random phase, spectral action, density balance equation for wave number-direction spectra. The equation's implicit assumption is that water depth and currents as well as the wave field itself vary on temporal and spatial scales that are much larger than the variation scales of a single wave. It includes some source term options for the surf zone and the wetting and drying of grid points.

1.4 Navy Coupled Ocean Data Assimilation (NCODA) System

The NCODA system was developed as an oceanographic version of the multivariate optimum interpolation (MVOI) technique widely used in operational atmospheric forecasting systems. Recently, NCODA was updated to a fully three dimensional variational assimilation system (NCODA 3DVar) but can operate in either capacity. NCODA includes 3D ocean analysis variables such as temperature, salinity, geopotential, and vector velocity components that are analyzed simultaneously. It can run in stand-alone mode but may be cycled with atmosphere and ocean models to provide updated initial conditions for the next model forecast in a sequential incremental update cycle. NCODA uses a volume formulation that permits several thousand observations located in the same region to be processed simultaneously. Observations include altimeter sea surface height (SSH) anomalies, SST, sea ice concentration, in situ SST data from ships and buoys, and temperature and salinity profile data from expendable bathythermographs (XBTs), conductivity, temperature and depth (CTDs) meters, and Argo floats. It makes full use of all sources of operational ocean observations and new ocean observing systems may be added as they become available. All observations are quality controlled through the NCODA_QC program (Cummings, 2005).

1.5 Earth System Modeling Framework (ESMF)

COAMPS and its support models are integrated through the Earth System Modeling Framework (ESMF). Funded by the Department of Defense (DoD) and NASA and developed by the National Center for Atmospheric Research (NCAR), the ESMF was

created as a high-performance, flexible software infrastructure to increase ease of use, performance portability, interoperability, and reuse in climate, numerical weather prediction, data assimilation, and other earth science applications. The software infrastructure allows various weather, climate, and data assimilation components to work together on an array of platforms, from laptops to supercomputers. ESMF software is component-based, representing models as collections of smaller modules that are coupled together. In ESMF, a component may be a physical domain or a function, such as a coupler or input/output (I/O) system. The framework provides tools for common modeling functions, as well as regridding, data decomposition, and communication on parallel computers.

1.6 Document Overview

The purpose of this User's Guide is to describe the setup and execution of the Coupled Ocean Atmosphere Mesoscale Prediction System, Version 5.0. COAMPS is continually undergoing enhancements and modifications. A validation of the atmosphere and ocean coupling was completed in 2010 (Allard et al., 2010). This document represents revision 2.0, which includes the addition of the SWAN wave model. Validation of COAMPS with SWAN is underway.

Please refer to Chen et al., (2003) for a comprehensive description of the basic equations, theory, and code of the COAMPS model. For NCOM, see Barron et al. (2006) and Martin et al. (2008a). For NCODA, see Cummings and Carroll, 2006. For SWAN, see the SWAN Technical Document (2010) at http://iod.ucsd.edu/~falk/modeling/swantech.pdf. WAVEWATCH III documentation may be found at http://polar.ncep.noaa.gov/mmab/papers/tn276/MMAB_276.pdf.

2.0 SETTING UP A SIMULATION

There are two platforms on which the user may set up the COAMPS system: 1) on the Grid Engine, which is the shared network of computers used to run jobs locally at NRL-SSC, or 2) at the DoD Supercomputing Resource Center (DSRC) at the Naval Oceanographic Office (NAVOCEANO). The quickest and most efficient way to run COAMPS is at the DSRC, where the program is allowed to automatically grab the input data from *NEWTON*, the DSRC's data archive server, and then place them in the proper directories specified by the **setup_navy_dsrc** script (Appendix A-3). If running COAMPS at the DSRC the directory structures for any user are the same. If running on the Grid Engine, COAMPS will grab the necessary files from *NEWTON* and send them to the directories specified by the user in **setup_nrlssc** (See Section 2.2.1.1 and Appendix A-1).

2.1 Initial Setup

When setting up an original model domain, locate the latest run script and namelist files on the new run system branch of the subversion repository (See Section 2.1.1 below). A COAMPS directory must be generated where the new domain's project will be created. After a COAMPS directory is established, execute the following commands to download the new scripts and namelists from the subversion control:

```
cd   [your COAMPS directory]
svn co http://coamps.nrlmry.navy.mil:8000/svn/run-coamps5/branches/ssc
run-coamps5
```

Several subdirectories (including ***jobs/***, ***projects/***, and ***scripts/***) have now been created in the COAMPS directory. The next step is to setup a project within the newly created directories. Create a name for the project, typically based on the geographic region, e.g. `[Adriatic]`, and execute the following commands from the COAMPS directory (`[region]` refers to the project name):

```
cd sscrun
cp –a jobs/area  jobs/[region]
cp –a projects/area projects/[region]
```

Several setup scripts have now been copied from the generic ***jobs/$area*** directory to the user's ***jobs/$area*** directory. Project namelist and OCARDS files have been copied from the generic ***projects/$area*** directory to the ***projects/$area*** directory.

2.1.1 COAMPS Subversion Repository

COAMPS developers at NRL routinely make changes, improvements, and bug fixes to the model, often concurrently. Therefore, they have created a COAMPS subversion repository (http://subversion.tigris.org/) (Collins-Sussman et al., 2007), whereby different versions of COAMPS and its complete developmental history are stored and available for user access. The subversion repository is accessible primarily to those individuals authorized by either Tim Campbell or Travis Smith at NRL-SSC. The primary COAMPS repository is located at NRL-Monterey. Those using COAMPS outside NRL-SSC and NAVOCEANO should send a request for repository access to Sue Chen at Sue.Chen@nrlmry.navy.mil. The internet address for the repository is http://coamps.nrlmry.navy.mil:8000/svn/run-coamps5/branches/ssc.

Those wishing to access the full repository available at NRL-SSC should send an email request to Tim Campbell (tim.campbell@nrlssc.navy.mil). Send Dr. Campbell an digitally signed email request and he will reply with an encrypted email containing a username and initial password. After receiving the initial password, go to https://www7320.nrlssc.navy.mil/svn/websvn and click on the "Change Your SVN Password" link to customize your password.

For those users local to NRL-SSC running COAMPS on the local network, a repository has been created at NRL-SSC that mirrors the NRL-MRY repository. No svn user account is required. To check out a copy of these scripts on the local network, the user should go to ***/u/COAMPS/src/coamps_sscrun_mirror***. Type in the command:

```
%>  svn co file:///u/COAMPS/src/coamps_sscrun_mirror sscrun
```

2.1.2 Setting and Checking Parameters

Only parameters specific to the setup of a new COAMPS simulation are mentioned in this User's Guide. There are many other namelist parameters that may be specified in both NCOM and COAMPS. Refer to the NCOM Version 4.0 User's Manual (Martin et al., 2008b) and the COAMPS User's Manual (Chen et al., 2003), respectively, for tables with descriptions of those parameters. SWAN model (The SWAN Team, 2010) documentation may be found at http://swanmodel.sourceforge.net/online_doc/swanuse/swanuse.html. The latest version of the WAVEWATCH III documentation is available at the NCEP website:
http://polar.ncep.noaa.gov/mmab/papers/tn276/MMAB_276.pdf.

2.2 Creating a Directory Structure

2.2.1 Customizing the setup_nrlssc Script

The **setup_nrlssc** script, located in the ***jobs/$area*** directory, is a COAMPS setup for the NRLSSC platforms such as PBS (Intel xeon nodes with the portable batch system), SGE (Intel xeon nodes with Sun grid engine batch system) and Mac (Apple MacBook Pro with no batch system) on the Grid Engine. It establishes data directories for the simulation, sets the platform-specific variables utilized in the **run_coamps** script, and creates platform-specific settings for sending output to stdout. It is invoked by the **run_coamps** script. IMPORTANT: All directories in the **setup_nrlssc** script must have correct read and/or write permissions set.

Several of the **setup_nrlssc** script commands must be manipulated in order to establish the necessary data directories for a new run. Instructions will proceed from the beginning of the script to the end. A sample **setup_nrlssc** file is shown in Appendix A-1. No input parameters are necessary. Table 1 summarizes the required global variables and those created in this script.

Table 1: Global variables required and created for setup_nrlssc.

Required variables	
Variable	**Description**
area	Name of simulation area/experiment.
batch_nprocs	Number of processors for a batch request.
cmdFile	Batch command file (with path).
cmdLog	Batch command log file (with path).
ddtg	Date-time-group of run.
interactive_option	Flag indicating a job is interactive.
jobDir	Path to ***jobs/*** directory (where **run_coamps** is invoked).
jobName	Name of batch job.
ksh_executable	Path to KornShell 93 executable.
platform	Name of platform (host machine).
USER	User name (environment).
wave_type	Wave model type (SWAN, WW3, or none).
Created variables	
Variable	**Description**
runDir	Area dDTG run directory.
prjDir	A project's area/experiment input file directory.
srcDir	Full network file system (NFS) path to the COAMPS source directory.
batch	Batch submission command.

*2.2.1.1 Local Grid Engine Setup of COAMPS (***setup_nrlssc***)*

When running COAMPS on the DSRC, the input data will automatically be pulled from *NEWTON*. When working on the Grid Engine, perform the following steps:

1. Open the **setup_nrlssc** script in the ***jobs/[PROJECT NAME]*** directory. The UNIX command `vi` can be used to open the script:

```
vi setup_nrlssc
```

2. Set the wall time, which is the actual time taken by a computer to complete a run from start to finish, in HH:MM:SS format. The wall time in **setup_ nrlssc** does not currently affect the user's position in the local queue. However, be sure to set the wall time sufficiently long so as to allow sufficient time for the simulation to finish.

```
typeset wallTime="12:00:00"
```

3. The queue on the Grid Engine, which is Torque, is always set to "standard".

```
typeset queue="standard"
```

4. Under the "**Local Variables**" heading, there are paths defined for the output and save directories, ***outDataDir*** and ***savDataDir***, respectively.

    a) The ***outDataDir*** houses the output data from the atmospheric, ocean and wave components of COAMPS. The directory that the user specifies, must exist (i.e., it will not be created.) Choose a storage location with enough memory space to house all the output data. Define an output directory where all output will be stored. Quotations must be used before and after the directory path. The following is an example of a path to the local COAMPS output data directory:

```
outDataDir = "u/COAMPS/scratch/$USER"
```

Output data for the run "Adriatic", for example, will be sent to the following directories:

Atmospheric data: ***outDataDir/Adriatic/atmos***
Ocean data: ***outDataDir/Adriatic/ocean***
Wave Data: ***outDataDir/Adriatic/wave/***
Ocean background SST tendency data: ***outDataDir/Adriatic/obkgd***

b) The save directory, ***savDataDir***, functions as storage for certain COAMPS data files. It is not currently being used in simulations but may be used in the future. Example:

```
savDataDir="u/COAMPS/data/$USER"
```

.

2.2.1.2 Selecting the Bin Directory Path

The ***srcDir*** is a full NFS path to the COAMPS bin directory. It points to the executables used in COAMPS. In `/u/COAMPS/src/` there are several sets of executables, but when a user downloads the scripts only the latest set of executables (with the latest DTG) will be used for the model and only the ESMF branch. ***SrcDir*** is a common directory for all COAMPS users at NRL-SSC.

For example, in order to choose the bin directory path in single precision mode, specify "r4" executable for single precision mode or "r8" executables for double precision mode:

```
srcDir = '/u/COAMPS/src/coamps4_esmf_r4_20111013'
```

or:

```
srcDir= '/u/COAMPS/src/coamps4_esmf_r8_20111013'
```

2.2.1.3 Creating the Path to Necessary Databases

The ***databaseDir*** houses input data such as bathymetry and terrain data files. This is a common directory for all users. Please do not alter it.

```
databaseDir=/u/COAMPS/input/database
```

2.2.1.4 Setting the Path for NOGAPS and ADP Input Files

The ***fandaDir*** is the path to the NOGAPS and ADP input files. The NOGAPS files are used for initial and boundary conditions (IC and BCs) while the ADP files contain data used in the atmospheric data assimilation. The NOGAPS and ADP **.tar** files needed for a particular project must be downloaded from the local archive to this exact directory. As a courtesy, several years of 1° NOGAPS and ADP data from *NEWTON* have been downloaded and stored locally in the ***/u/COAMPS/input*** directory for NRL-SSC use.

```
fandaDir=/u/COAMPS/input/fanda
```

NOGAPS and ADP files are archived locally at NRL-SSC.

Note: Access to NOGAPS and ADP data not found in ***/u/COAMPS/input/fanda*** may be accessed by contacting Travis Smith or Tim Campbell at NRL. Please send an email request to travis.smith@nrlssc.navy.mil with the time frame of the data needed.

2.2.1.4.1 Downloading from the *NEWTON* Archive (if the USER has permission)

Access to the *NEWTON* archive requires group permission from NRL-MRY Code 0024. Email james.doyle@nrlmry.navy.mil to request permission to access it. Also, a user may contact Tim Campbell (tim.campbell@nrlssc.navy.mil) or Travis Smith (travis.smith@nrlssc.navy.mil) if data needs to be downloaded.
NOGAPS data are located on *NEWTON* in:

```
/u/home/hodur/masfnmc
```

ADP data are located on *NEWTON* in:

```
/u/home/hodur/adp
```

The NOGAPS **.tar** files must then be extracted in the ***fandaDir***. The `tar –xf` command extracts each **.tar** file, which then automatically produces the ***f$dtg*** directories used by the model.

Each individual ADP **.tar** file must be placed in a directory named ***adp$dtg*** ($dtg being of the form YYYYMMDDHH). NOTE: *This is not done automatically when extracting each ADP file*. First create the ***adp$dtg*** directory in the ***fandaDir***, place the ADP **.tar** file in the ***adp$dtg*** directory, and then execute the extraction using both gunzip and tar xf.

For every DTG, grab each 12-hourly NOGAPS 00 and 12 hr file and each 00 and 12 hr ADP file. The six-hour NOGAPS files are also included in the 00 and 12 hr NOGAPS files.

2.2.1.5 OCNQC Data Path Selection

The ***ocnqcDir*** houses the OCNQC data used by NCOM for the ocean analysis and data assimilation. The OCNQC is being used in the ocean data assimilation (MVOI or 3DVAR NCODA) and to produce an SST field for the bootstrap step of the atmospheric model. The bootstrap step is just one time step of the atmospheric model that creates the initial atmospheric analysis fields. The bootstrap step uses NCODA sea surface temperatures (SSTs) for these initial fields only in a fully coupled COAMPS run. All subsequent time steps following the bootstrap step use NCOM SSTs. An atmospheric-only run still needs OCNQC data to produce an SST field.

For NRL-SSC, all the OCNQC files back to 2008010100 are archived locally through the path:

```
/u/prob/ncoda/data/navoqc/navoqc_public/
```

If OCNQC data is necessary before 2008, it will need to be downloaded to the directory:

```
/u/COAMPS/input/ocnqc
```

The NAVOCEANO restricted OCNQC data pathway is:

```
ocnqcRstrctDir=/u/prob/ncoda/data/navoqc/navoqc_restrct
```

Restricted data access will require group permissions to ***otdata***. Please contact Pete Spence (Peter.Spence.ctr@nrlssc.navy.mil) to determine whether restricted data is available for the user's particular region and time frame.

Note: OCNQC data access requires permission from NRLSSC. The user may contact Tim Campbell (tim.campbell@nrlssc.navy.mil) or Travis Smith (travis.smith@nrlssc.navy.mil) if data needs to be downloaded.

2.2.1.6 Global NCOM Data Retrieval

The **setup_nrlssc** script automatically locates by DTG the global NCOM data files used for open ocean boundary conditions. These are locally stored at NRL-SSC.

```
gncomDir=/u/NCOM
```

2.2.1.7 Wave Data Storage

The ***gwaveDir*** directory points to SWAN boundary condition files that have been generated for SWAN by running WW3. WW3 global data is not stored at NRL-SSC, so the WW3 model must be run separately to generate boundary conditions (BCs) for SWAN. The ***gwaveDir*** directory can be specified to any directory pointing to the SWAN BC files.

```
gwaveDir=/u/WW3
```

2.2.1.8 Setting Paths to Model Output Data Files

Output data file paths for the atmosphere, ocean, and wave models include ***atmosDir***, ***oceanDir***, ***obkgdDir***, ***waveDir,*** and several other directories. The output data will go in the following directories depending on the user's specifications for ***outDataDir*** earlier in the script.

```
dataDir=$outDataDir/$area
atmosDir=$outDataDir/$area/atmos
oceanDir=$outDataDir/$area/ocean
waveDir=$outDataDir/$area/wave
obkgdDir=$outDataDir/$area/obkgd
wbkgdDir=$outDataDir/$area/wbkgd
cplrDir=$outDataDir/$area/cplr
```

2.2.2 Customizing the setup_area Script

In the ***jobs/$area*** directory, the **setup_area** file is used to setup specific run parameters for the simulation. Within this script, the user must specify both the site (DSRC or local) and the platform (*DAVINCI, EINSTEIN*, or local machine) on which COAMPS is being run. For example:

`site=navy_dsrc` (for DSRC usage)
`site=nrlssc` (for local NRL-SSC usage)

`platform=` `davinci` or `einstein` (for DSRC usage)
`platform=pbs(` for running COAMPS on the Grid Engine at NRL-SSC)

Table 2 summarizes the global variables created and required while running the script. Step-by-step instructions on the various script commands requiring user intervention proceed from the beginning of **setup_area** script (see Appendix A-2) to the end. The script is subject to change based on factors that impact the model run, such as the frequency of the input data specified (3- or 6-hourly) or the number of processors.

Table 2: Global variables within the setup_area script.

Created variables	
Variable	**Description**
area	Name of simulation area/experiment.
atmos_analysis_type	Atmospheric analysis type (MVOI or 3DVar).
atmos_nproc(x,y)	Number of tiles in *x*- and *y*-direction for atmospheric model.
atmosa_nprocs	Number of processors for atmospheric analysis (NAVDAS).
atmosa_nthreads	Number of OpenMP threads for atmospheric analysis (NAVDAS).
cmdFile	Batch command file (with path).
cmdLog	Batch command log file (with path).
couple_A2B	Coupled forecast with model A-to-model B exchange turned on/off.
coupled_execution_mode	Coupled forecast model execution mode (sequential or concurrent).

Table 2: Global variables within the setup_area script.	
Created variables	
Variable	**Description**
fcst_length	Number of hours for forecast.
jobName	Name of batch job.
ocards & xcards	Names OCARDS and XCARDS files (in ***projects/*** directory).
ocean_analysis_type	Ocean analysis type (MVOI or 3DVar).
ocean_nproc(x,y)	Number of tiles in *x*- and *y*-direction for ocean model.
oceana_nprocs	Number of processors for the ocean analysis (NCODA).
oceana_nthreads	Number of OpenMP threads for the ocean analysis (NCODA).
platform	Name of platform (corresponds to setup_${platform}.sh).
site	Location at which COAMPS is being run (DSRC or local machine).
update_cycle	Number of hours for hindcast/forecast update cycle (6 or 12 hr).
wave_nprocs	Number of processors for wave model.
wave_type	Wave model type (SWAN, WW3, or none).
Required variables	
jobDir	Path to job directory (where **run_coamps** is invoked).

2.2.2.1 Choosing the Update Cycle

The *update_cycle* parameter (default = 12) sets the hindcast/forecast update cycle and defines the time in which data assimilation (atmosphere and/or ocean) will be executed. It can be set for either 6 or 12 hours. A six hour update cycle is useful only for the atmospheric model and if ADP data is available in 6 hourly intervals rather than 12 hourly intervals in the input data directory. A twelve hour update cycle is recommended for the ocean and atmosphere update cycle for hindcast simulations. Future updates are planned in which both the atmosphere and ocean update cycles would be set separately.

```
update_cycle=12
```

2.2.2.2 Setting the Forecast Time Length

The number of forecast hours is controlled by setting the *fcst_length* parameter in the **setup_area** script (Appendix A-2**)**. Generally, hindcast runs will use a 12 hour forecast length with a 12 hour update cycle due to existing data. For real time, fully-coupled air/ocean/wave runs, a maximum of 96 hours can be simulated (forecast starting at 00Z) or a maximum of 84 hours (forecast starting at 12Z) depending on global NCOM forecast field availability. Even though the global NCOM only has output up to 96 hr, the coupled model can run beyond 96 hrs by holding the NCOM boundary conditions fixed past 96

hours. This functionality is set by the **setup_nl_hostnl** namelist parameter *host_fcstlen*. It is currently set at 96 (See Section 2.3.2.3).

.

```
fcst_length=12
```

2.2.2.3 Atmosphere and Ocean Coupling

The **setup_area** (Appendix A-2) script specifies the type of coupling between the atmosphere, ocean, and wave models. With the addition of the SWAN model, there is the potential for up to six-way coupling. The parameters *couple_a2o* and *couple_o2a* specify a coupled forecast with atmosphere-to-ocean exchange or ocean-to-atmosphere exchange, respectively. The atmosphere/wave parameters, *couple_w2a* and *couple_a2w*, and the ocean/wave parameters, *couple_w2o* and *couple_o2w*, specify wave coupling options with the SWAN and WAVEWATCH III models. The *couple_w2a* parameter is currently still experimental.

The following possibilities exist, along with a table of possible combinations and commands in Table 3:

- For a fully coupled "atmos/ocean" run only, set *couple_a2o* and *couple_o2a* to "t". Set parameters involving the wave model to "f".
- To specify a one-way coupled run with no ocean feedback to the atmosphere, but still provide COAMPS wind forcing to NCOM, set *couple_a2o* to "t" and *couple_o2a* to "f".
- For full air/sea/wave coupling set *couple_w2a* and all other parameters to "t". While it currently works in all versions it is still experimental. Validation will start in FY2012.
- To turn off ocean feedback to the wave model, set *couple_o2w* = f.
- To turn off wave feedback to the ocean model, set *couple w2o* = f.
- Default values are all set to "t".

couple_a2o="t"
couple_o2a="t"
couple_a2w="t"
couple_w2a="t"
couple_o2w="t"
couple_w2o="t"

Table 3: Current COAMPS coupling capabilities.

Coupling	***atm2ocn***	***ocn2atm***	***atm2wav***	***wav2atm***	***ocn2wav***	***wav2ocn***
Six-way fully-coupled air/ocean/wave	t	t	t	t	t	t

Coupling	***atm2ocn***	***ocn2atm***	***atm2wav***	***wav2atm***	***ocn2wav***	***wav2ocn***
Two-way coupled air/ocean only	t	t	f	f	f	f
One-way coupled air/ocean only	t	f	f	f	f	f
Two-way coupled ocean/wave only*	f	f	f	f	t	t
Atmosphere only (use **run_area** option –f2)	f	f	f	f	f	f
Ocean only*(use **run_area** option –f3)	f	f	f	f	f	f
Wave only*(use **run_area** option –f4)	f	f	f	f	f	f

*- This forecast requires special instructions.

Running the Ocean only and Wave only forecasts necessitates providing a wind input file for forcing. This wind input file will be generated if COAMPS has been previously run for that particular region and time and if the wind output was saved. When choosing **run_area** option *-f3* or *-f4*, go to **setup_nrlssc** and change ***$atmosDir*** to the directory that contains the wind output that the user will need to generate a wind input file when running ocean only or wave only. A wind input file will be generated for forcing.

NOTE: The parameters being coupled between each model are specified in the **setup_esmf_config** script found in the **/projects/$area** directory. Refer to Section 2.3.2.6 for information on the coupling variable in **setup_esmf_config.**

2.2.2.4 Wave Model Type

There are two types of wave models that may be coupled into COAMPS, the Simulating Waves Nearshore model (SWAN) and WAVEWATCH III (WW3). SWAN may be run in wave-only forecast mode.

```
wave_type=swan
```

2.2.2.5 Coupled Execution Mode

The *coupled_execution_mode* is set to "sequential" when running COAMPS on a small number of processors as on the Grid Engine. When in sequential mode, this parameter will run a coupled forecast with the atmosphere, ocean, and wave models running on the same set of processors. The total # of processors = # of atmospheric processors = # of ocean processors = # of wave processors. When running COAMPS on the DSRC with a large number of processors, set this parameter to "concurrent" so that atmosphere, ocean, and wave components are run on different sets of processors. The total # processors = # atmospheric processors + # ocean processors. For DSRC jobs with smaller numbers of

processors, it is recommended that *coupled_execution_mode* be set to "sequential". The default is "sequential".

```
coupled_execution_mode=sequential
```

2.2.2.6 Selecting the Number of Processors

COAMPS can be run with any number of processors. A user may have to test out different combinations to achieve maximum efficiency. Set the appropriate number of processors depending on the location of the run, whether it is the Grid Engine or the DSRC. The product of *nprocx* and *nprocy* equals the total number of processors being used. For this example, the total number of processors is 2(4) = 8. Please see the "`batch=qsub`" command section in the **setup_nrlssc** script (Section 2.2.1.3; Appendix A-1) for running COAMPS on the Grid Engine.

The wave model only supports 1D decomposition. Therefore, the grid cannot be decomposed into both x and y to send smaller grid sections to a different number of processors. Depending upon the grid size, the user must be careful not to allot too many or too few processors for SWAN. Typically, one processor needs at least five grid points to run properly.

```
atmos_nprocx=2; atmos_nprocy=4
ocean_nprocx=2; ocean_nprocy=4
wave_nprocs=8
```

It is not possible to change the number of processors for SWAN from one forecast to another. For example, a user runs a hindcast from 2012010100 to 2010010500 using a 12 hour update cycle. He/she then chooses 32 processors for SWAN for a 12 hr 2012010100 forecast and the forecast completes its run. For the next DTG, 2012010112, the user *must* use 32 processors.

2.2.2.7 Atmospheric Analysis Options

The parameter *atmospheric_analysis_type* can be either MVOI (`mvoi`) or 3D variational analysis (`3dvar`). The default is `3dvar`. It is advisable to choose 3DVar even when MVOI is available, as NAVDAS has proven to be much more accurate.

```
atmos_analysis_type=3dvar
```

The parameter *atmosa_nprocs* is used to specify the number of processors to run NAVDAS for the atmospheric assimilation (the NAVDAS or MVOI option is specified in **setup_nl_coamnl**; see Appendix B-1). The minimum number of processors to run NAVDAS is four.

```
atmosa_nprocs=8
```

The parameter *atmosa_nthreads* is the number of OpenMP threads for the atmospheric analysis. It will be capped to *num_physical_cpus_per_node* (set in **setup_<site>**).

```
atmosa_nthreads=8
```

2.2.2.8 Ocean Analysis Options

There are two ocean analysis options from which to choose. As with the atmospheric analysis options, the user must select either MVOI (`mvoi`) or 3DVar (`3dvar`). The default option for *ocean_analysis_type* is `3dvar`. Refer to Section 2.3.2.5 for choosing the namelist parameters available for 3DVar in **setup_nl_oanl**.

```
ocean_analysis_type=3dvar
```

For the parameter *ocean_analysis_update_opt* in the **setup_area** script, the options for ocean assimilation are *update* or *relax*. For the update option, the analysis variable increment is applied to the model restart:

```
ocean_analysis_update_opt=update
```

For the relax option, the relaxation intervals are set by *rlax_ts* and *rlax_uv* for temperature, salinity and currents. The rlax_ts is a timescale for relaxation of deep T and S (in hours). It is typically set at 6.0. If it is set equal to 0, a spatially variable 3D field of relaxation weights is read in from an input file (that must be provided), which allows different specifications of the relaxation timescale at each model grid point. Stronger relaxation would be necessary near open boundaries and weaker relaxation near the coast. If *indlxts* is set at 3 or 4 it specifies the insertion period for T/S increments. The parameter *rlax_uv* is the timescale for relaxation of deep U and V (in hrs). It is also set to 6.0 unless otherwise specified. The analysis variable increment is applied over the relaxation period, specified in **setup_nl_oparm**.

The number of processors for ocean analysis is set with the parameter *oceana_nprocs*.

```
oceana_nprocs=16
```

The parameter *oceana_nthreads* is the number of OpenMP threads for ocean analysis. It will be capped to *num_physical_cpus_per_node* (set in **setup_<site>**).

```
oceana_nthreads=8
```

2.2.2.9 Global Atmospheric Input Files

The user may choose to use 0.5° (NOGAPS64) or 1° (NOGAPS36) NOGAPS files using the **setup_area** parameter *host_atmos_type*. Generally the 1° NOGAPS is only available in the 36 file name convention and the 0.5° is available only in the 64 character convention. Those wishing to use 0.5° NOGAPS will need to contact Travis Smith (travis.smith@nrlssc.navy.mil) or Tim Campbell (tim.campbell@nrlssc.navy.mil) to request the data, since NRLSSC currently only stores 1° NOGAPS.

```
host_atmos_type=nogaps36 -for one degree NOGAPS.
host_atmos_type=nogaps64 -for 0.5 degree NOGAPS.
```

2.2.2.10 OCARDS and XCARDS Files

An OCARDS file allows the user to create hourly (or any time interval) flat file output for several important variables on a specified grid, such as wind, wind stress, and heat flux. The **OCARDS.default** file may be used to output certain COAMPS flat files. Any output not generated in the **OCARDS.default** file may be chosen by the user through changing **OCARDS.default** to **OCARDS.$area**, which is generated by the **gen_ocards.ksh** script located in the ***/projects*** namelist directory. Copy it to the ***projects/$area*** folder for the user's specific project area to generate user-selected atmospheric flat file output. XCARDS files are not used at this time in the coupled system. For additional assistance with using the OCARDS generation script, please contact Travis Smith (travis.smith@nrlssc.navy.mil).

The following commands name the OCARDS and XCARDS files in the ***projects/*** directory.

```
ocards=OCARDS.$area
xcards=XCARDS.$area
```

2.2.3 Customizing the setup_navy_dsrc Script

The **setup_navy_dsrc** script automatically creates a setup for the user to access *DAVINCI* or *EINSTEIN*, two of the Navy's DSRC IBM P5+ systems for the queuing and running of jobs. The script sets variables to directly pull the ADP, OCNQC, NOGAPS, and global NCOM files from *NEWTON* and put them in directories created within the script. Appendix A-3 contains the entire **setup_navy_dsrc** script.

2.2.3.1 Specifying the Project Account Number

When running on the DSRC, it is necessary to enter a project account number (*projAcct*) to which the particular COAMPS run will be charged. The following command in the **setup_navy_dsrc** must be changed to reflect the correct account number.

```
typeset projAcct="NRLSS060"
```

The Portable Batch System (PBS; see Section 2.2.3.3) directive for *projAcct* is:

```
#PBS -A ${projAcct}
```

2.2.3.2 Specifying the COAMPS Run Time

The *wallTime* is the amount of time a user may request for a COAMPS run. For example, if a 12 hour forecast takes 45 minutes then the user should request an hour of run time in order to limit the time that the job is waiting in the queue. The format for *wallTime* is "HH:MM:SS", so one hour would be:

```
typeset wallTime="01:00:00"
```

The batch directive for *wallTime* is:

```
#PBS -l walltime=${wallTime}.
```

2.2.3.3 Checking the Status of a DSRC COAMPS Run

The user may check the status of the jobs submitted to the DSRC by using the PBS commands. PBS allocates batch jobs among the available DSRC platforms. The following commands shown in Table 4 are evoked by simply typing them at the command prompt.

Table 4: Portable Batch System (PBS) commands for checking the status of a COAMPS run on the DSRC.

Command	Description
`qstat`	Shows the status of all running and pending jobs.
`qstat –u [username]`	Shows the status of current jobs.
`qpeek [job ID number]`	Displays the progress of the COAMPS run.
`qdel [job ID number]`	Kills a job in progress.

2.2.3.4 Setting the Input Data Directory Path

All input data will be brought down from *NEWTON* and added to ***inpDataDir***, which is the path to the local COAMPS input data directory. It houses NOGAPS, ADP, and OCNQC data.

```
typeset inpDataDir="$scratchDir/COAMPS/data"
```

2.2.3.5 Setting the Output Data Directory Path

The outDataDir path needs to be changed to designate the user's preferred local output directory for all COAMPS atmosphere, ocean, and wave data output.

```
typeset outDataDir="$scratchDir/COAMPS/data"
```

2.3 Setting up Atmospheric and Oceanic Namelists

The following namelist files are necessary for the COAMPS setup. These namelist files are located in ***[COAMPS directory]/sscrun/projects/$area***.

setup_esmf_config
gridnl.atmos
gridnl.ocean
gridnl.wave
setup_nl_soilnl
setup_nl_omnl
setup_nl_setupl
setup_nl_postnl
setup_nl_aeronl
setup_nl_hostnl
setup_nl_atmosnl
setup_nl_coamnl
setup_nl_mvoinl
setup_nl_omnloff
setup_nl_oparm
setup_nl_oanl
setup_ww3_grid
setup_ww3_multi
setup_ww3_outf
setup_ww3_outp
setup_ww3_strt
gen_ocards.ksh
OCARDS.default

2.3.1 COAMPS Atmospheric Namelist Setup

2.3.1.1 Choosing the Atmospheric Nests and Levels

After generating the atmospheric grid setup for COAMPS using COAMPS On-Scene (COAMPS-OS) (See Section 2.4 for stepwise instructions on using the COAMPS-OS® GUI on *FST1* at NRL-SSC), transfer the grid setup (**gridnl**) to **gridnl.atmos** (Appendix C).

- *nnest*- The number of nests is specified in the parameter *nnest*. This parameter may be changed to any number of nests less than the total number of nests originally specified without making changes to the namelist. Rerun the atmospheric analysis when changing the number of nests to ensure proper functionality.
- *kka*- The number of atmospheric levels, *kka*, may be set to 30, 40, or 60 levels. The *kka* value is currently set to 60. The correct sigma level specifications (through *dsigma*) are automatically chosen depending on the value of *kka* in **gridnl.atmos.**
- *llmove*- This is used for moving nests in COAMPS-TC (See Section 3.5.)

2.3.1.2 Selecting Boundary Condition and MVOI Levels (setup_nl_atmosnl)

The atmospheric namelist, **setup_nl_atmosnl** (Appendix B-6), is important for setting up the number of MVOI and boundary condition (BC) levels.

- *lm*- The number of atmospheric analysis pressure levels. Currently, *lm* is automatically set whether MVOI or NAVDAS is used for the atmospheric analysis.
- *lmbc* - The number of boundary condition levels. It changes depending on the NOGAPS data of the hindcast. For the Adriatic test case of February 2003, *lmbc* was set to 26 based on the number of NOGAPS BC levels. This parameter is automated to read the NOGAPS files and determine the number of levels necessary to run COAMPS.
- *ingres* and *jngres* - These automated parameters set the global NOGAPS grid sizes that the atmospheric analysis will utilize. For 1° NOGAPS, *ingres* is set to 380 and *jngres* is set to 181. For 0.5° NOGAPS, *ingres* = 720 and *jngres* = 361.
- *maxout* - The maximum number of atmospheric output files that are generated by the model. If the OCARDS file that is created for atmospheric output has more than 12,000 files output, *maxout* must be increased.

*2.3.1.3 Setting COAMPS-Specific Parameters (***setup_nl_coamnl***)*

In the **setup_nl_coamnl** script, most of the parameters are set to their optimum default values (See Appendix B-1. However, several parameters have been added or changed.

- *delta* - One of the more important parameters, *delta,* specifies the atmospheric model time step for the course nest (in seconds). It is useful in tweaking the model run and customizing it to the user's needs. The inner nest uses a 3:1 time step ratio. The *delta* default is set to 90 seconds. The atmospheric model will calculate a time step based on the resolution of the atmospheric nests. If the time step is set too large, a warning message will appear in the **log.aanalysis** file and a time step will be recommended for the atmospheric model.
- *kgetbc* and *itauin* - Used by the atmospheric analysis and forecast to create and use boundary conditions (3, 6 or 12 hourly, depending on the availability of NOGAPS).
- *lanalbc* - Specifies whether analysis fields or forecast fields are to be used in generating the atmospheric boundary condition from NOGAPS data. Analysis fields typically are preferred for hindcasts; however, the analysis fields may not be available for the DTGs of interest. Forecast fields are always available for all DTGs.
- *l2way* - Allows for feedback between the child atmospheric nest back to the parent domain. It is currently set to "t".
- *nradtyp* - Refers to the radiation scheme type used by the atmospheric portion of COAMPS. Option 1 uses the original COAMPS radiation scheme. A default value of 2 uses the Fu-Liou radiation scheme.
- *idbms* - specifies the format for input and output files. It uses 36- and 64-character flat files for the creation of boundary conditions, such as 0.5° NOGAPS. Set *idbms* to 2. The *idbms* options are:

1. Read old NOGAPS and write old COAMPS files (36 characters, sequential),
2. Read new NOGAPS and write new COAMPS files (64 characters, direct access),
3. Read new NOGAPS and write old COAMPS files (36 characters, sequential),
4. Read old NOGAPS, new COAMPS (*iupd*=2), and write new COAMPS files (64 characters, direct access), or
5. Read old NOGAPS, old COAMPS (*iupd*=2), and write new COAMPS files (64 characters, direct access).

2.3.2 NCOM Ocean Namelist Setup

*2.3.2.1 Ocean Grid Parameter Setup (***gridnl.ocean***)*

After generating the ocean grid namelist (**gridnl**), transfer the grid information to **gridnl.ocean**.

- *nnest*- The number of ocean nests. As with the atmospheric **gridnl** namelist, the nest number may be changed to a number less than the original setup. For example, if the original setup on RELO NCOM was for two nests, changing *nnest* to 1 will not require any additional changes to the namelist.
- *kkom* – The total number of ocean levels. The default is 50.
- *kkosm* – The total number of sigma levels.
- *delx and dely* - The x- and y- resolution of the ocean nests in meters.

*2.3.2.2 Ocean Parameter Setup (***setup_nl_oparm***)*

Alter the following parameters in **setup_nl_oparm** ocean namelist (See Appendix B-2):

- *dti_base* - The time step parameter for ocean nest 1. The time step for ocean nest 2 will automatically follow a 3:1 ratio to the nest 1 time step.
- *indiag* - This parameter prints out diagnostics to the screen. Change *indiag* to 1 for more diagnostics.
- *indriv* – This is the river input parameter. It is set to 1 for "on". Change *indriv* to 0 to turn off river input.
- *indzk* - The parameter that controls the mixing scheme. A value of 4 for *indzk* uses the Mellor-Yamada 2.5 (MY2.5) mixing scheme and a value of 5 uses MY2.5 with the addition of the Kantha and Clayson (2004) formulation when wave forcing is also invoked. If running COAMPS with a wave model, change *indzk=4* to *indzk* =5.
- *tidpot* – Setting tidpot to "true" turns on the local tidal potential. Tidal potential forcing is currently provided for constituents K1, O1, P1, Q1, K2, M2, N2, S2, MF, and MM. Constituents must be specified in an input file. Default is 'true'.

Note: Tidal potential forcing is needed if the user is running with tides in a large, deep water domain. It is irrelevant in shallow, coastal domains. Set to 't' for deep ocean domains.

- *feedbk* – This parameter allows for feedback of the child oceanic nest back to the parent domain (temperature and salinity).
- *indtide* - specifies tidal forcing at open boundaries. Choose 1 for "yes" and 0 for "no". Set *indtide* to 0 if creating boundary conditions from a previous NCOM run that already uses tides.
- *irs_out* – The user may choose to have an NCOM restart file output more than once if running a forecast. It will output a restart file at the interval set in '*out*' (1).
- *out* - There are 12 output control parameters for native NCOM files. Values are hourly. There are several output control options available as summarized in Tables 5, 6 and 7:

Table 5: Output control options (*outo*).

outo	Description	Default Values
1	Frequency for output of restart file (h).	12
2	Frequency for output of 3D fields (h).	1.0
3	Frequency for output of surface fields (h).	1.0
4	Frequency for output of NFS restart file (h).	${update_cycle}
5	Frequency for output of kinetic energy (h).	0.0
6	Frequency for output of values at single pts (h).	0.0
7	Frequency for output of transports (h).	0.0
8	Frequency for output of water-mass volumes (h).	0.0
9	Frequency for output of budgets (h).	0.0
10	Frequency for output of wave forcing (h).	0.0
11	Not currently used.	0.0
12	Not currently used.	0.0

Table 6: Output control for 2D surface fields output file (*iouto*).

iouto	Description	Default Values
inde2	Include surface elevation: =0 no, =1 yes.	1
indvb2	Include barotropic transport: =0 no, =1 yes. **Note:** = (depth-ave *u* and *v* velocity)*(depth).	1
indv2	Include surface velocity: =0 no, =1 yes.	1
indt2	Include surface temperature: =0 no, =1 yes.	1

iouto	Description	Default Values
inds2	Include surface salinity: =0 no, =1 yes.	1
inda2	Include surface wind stress: =0 no, =1 yes.	1

Table 7: Output control for 3D surface fields output file (*iouto*).

iouto	Description	Default Values
inde3	Include surface elevation: =0 no, =1 yes.	1
indvb3	Include barotropic transport: =0 no, =1 yes. **Note**: = (depth-ave *u* and *v* velocity)*(depth).	1
indv3	Include 3D *u* and *v* velocity: =0 no, =1 yes.	1
indw3	Include 3D vertical velocity:=0 no, =1 yes.	1
indt3	Include 3D temperature: =0 no, =1 yes.	1
inds3	Include 3D salinity: =0 no, =1 yes.	1
inda3	Include surface atm forcing: =0 no, =1 yes. **Note:** This includes surface wind stress, solar, net surface heat flux (longwave + latent + sensible), and evaporation - precipitation.	1

NOTE: The native NCOM output control is specified in the **setup_nl_oparm** namelist.

2.3.2.2.1 Forcing NCOM with Winds from a Previous COAMPS Run

Occasionally it may be necessary to force NCOM from a previously coupled forecast run, such as when running NCOM ocean grid simulations at very high resolutions. After completing an initial coupled run at a lower resolution, that lower resolution grid would be used to create boundary conditions for a new run with a higher resolution NCOM grid. Additionally, atmospheric forcing from the lower resolution NCOM run may also be used to force the higher resolution NCOM grid. When using a lower resolution run to create the new boundary conditions, the native NCOM output must be saved (**out3d** fields). The user may also use the *osflx* files that are output from NCOM to produce wind and heat flux forcing for another run. As shown in Table 8 below, the user can specify option 1 for each of the parameters to read the *osflx* input data files from the previous run in order to create the forcing.

In order to generate native NCOM **out3d** output, specify the following in **setup_nl_oparm**, found in the ***/projects*** directory:

'`outo !2`' =1.0, which is frequency for output of 3D fields (h) (See Table 5).

After an initial lower resolution NCOM run is complete, follow these directions:

1. Setup a new COAMPS run with the higher resolution grid for NCOM.
2. Create boundary conditions for the higher resolution NCOM grid. The host will no longer be global NCOM, but the lower resolution NCOM grid run just completed. In the ***/projects*** directory, change **setup_nl_hostnl** from the following settings:

```
   nrlssc)
typeset host_idtype=3
if (( $ymd < 20080601 ))
then
  typeset host_run="glb8_2f"
else
  typeset host_run="glb8_3b"
fi
typeset host_addyr=t
typeset host_dsogrd="/u/NCOM/glb8_2a/input"
typeset host_dsodat="/u/NCOM/${host_run}/nc"
typeset host_gclose="trpl"
;;
```

to these settings:

```
  nrlssc)
typeset host_idtype=2
typeset host_run="none"
typeset host_addyr=f
typeset
host_dsogrd="/u/COAMPS/scratch2/tasmith/Adr06valid/ocean"
typeset
host_dsodat="/u/COAMPS/scratch2/tasmith/Adr06valid/ocean"
typeset host_gclose="none"
;;
```

The parameter *host_idtype* will need to be changed from 3 to 2 to indicate that the BCs will be generated from RELO NCOM **out3d** files. The *host_run* parameter will need to be set to 'none', *host_addyr* will be set to 'f', *host_dsogrd* and *host_dsodat* now will be in the ***oceanDir*** where output is being stored, and *host_gclose* will need to be set to 'none'.

3. Change *host_tauinc* as specified in **setup_nl_hostnl** to the time interval specified by the user in the **setup_nl_oparm** for the output interval of the **out3d** files.

The ocean setup program for the new run will then generate an open boundary condition (opnbc) file from the lower resolution NCOM run that will be stored as usual in the ***oceanDir***.

To use atmospheric forcing from the lower resolution NCOM run, follow these steps:

1. In **setup_nl_oparm** in the ***/projects*** directory, change the parameters in Table 8 to specify COAMPS flat file forcing. To use the *osflx* files from NCOM to produce forcing, change the default values to 1,1,1,1,1, and 1. For coupling in the ESMF framework, all values should be 0.

Table 8: Surface forcing options and parameters (isbco, sbco).

Parameter	Option	Description	Default Values
indsbc	1	Atm forcing:=0 none (all turned off); =1 turned on.	1
indatp	2	Surface atm pressure:=0 none; =1 use *osflx* input data file; =2 use coupled atm model; =3 use COAMPS native grid files.	3
indtau	3	Wind stress: =0 none; =1 use *osflx* input data file;=2 use coupled atm model; =3 use COAMPS native grid files.	3
indsft	4	Surface heat flux: =0 none; =1 use *osflx* input data file; =2 use coupled atm model; =3 use COAMPS native grid files;=4 calc w/ bulk formula from COAMPS fields;=5 calc w bulk formula from data in *osflx* input file.	4
indsfs	5	Surface salt flux: =0 none; =1 use *osflx* input data file=2 use coupled atm model; =3 use COAMPS native grid files; =4 calc w/ bulk formula from COAMPS fields; =5 calc w/ bulk formula from data in osflx input file.	4
indsol	6	Solar flux: =0 none; =1 use *osflx* input data file; =2 use coupled atm model; =3 use COAMPS native grid files.	3

2. In **setup_nl_omnl** in the **/projects** directory, the user specifies the time interval and atmospheric nest required for the atmospheric forcing of the new NCOM grid:

- *inesta* - Specifies the atmospheric grid number that will be used to force NCOM. Please make sure that the grid specified covers the entire NCOM grid being forced. For example, if a user has three atmospheric grids from the lower resolution NCOM coupled forecast run, and the highest resolution atmospheric grid, grid 3, can completely cover the new high resolution NCOM grid, choose *inesta*=3.
- *dtcyc*- Specifies the length of the atmospheric update cycle from the lower NCOM resolution coupled simulation.
- *dtf* and *dtt* - Specify the interval of atmospheric forcing on the ocean model. Hourly forcing (= 3600) is the default. However, the user will need to make sure that hourly output for the atmospheric model is specified in the **OCARDS** file. The forcing is not equipped to handle atmospheric forcing intervals of less than 1 hour (3600 s).

3. Specify the location of the atmospheric flat files from the original run in ***atmosDir*** as designated in **setup_nrlssc** in the ***/jobs*** directory.

4. When forcing NCOM with the lower resolution NCOM coupled forecast atmospheric fields, it is necessary to turn the *o2a* and *a2o* coupling in **setup_area** in the ***/jobs*** directory to false (*couple_a2o*=f, *couple_o2a*=f).

5. The coupled model executables can still perform an ocean only (not coupled) forecast using COAMPS forcing from a previous run. To do so, use the following options in the **run_area** script when starting a new run with a higher resolution NCOM grid:

```
../../scripts/run_coamps -o1 -f3 2006020100
```

   The **-o1** option will complete the ocean setup as usual and will produce opnbc files from the previous run. The **-f3** option specifies an ocean only forecast and will use the COAMPS atmospheric files from the previous run to force the new run.

6. In the ***run/[DTG]*** directory, the log file **log.oforecast**' will be used for an ocean only forecast.

2.3.2.3 Ocean Input/Output Host Setup (setup_nl_hostnl)

Two parameters of note may be changed in the **setup_nl_hostnl** script (see Appendix B-7).

- *host_tauinc* – This parameter specifies the ocean boundary conditions. Depending on the availability of global NCOM data, *host_tauinc* can be specified for 3, 6, or 12 hour increments. The global NCOM archives generally keep 6 hourly data available.
- *host_run* – It specifies the host that houses the global NCOM input and output data used in the generation of the oceanic nest boundary conditions.

- *host_fcstlen* – This parameter sets length of the host forecast. When set > 0, it defines a minimum length in hours of the available and required host forecast. For boundary condition processing, if the setup cannot find a set of host files for a time past that minimum, the setup persists the last good set of data until the end of the NCOM forecast (by writing the same data labeled with a time at the end of the NCOM run). There is a constraint in the coupled model by the global NCOM forecast length of 96 hours at 00Z each day for a real-time run. However, a real-time forecast beyond 96 hours is possible by holding the NCOM boundary conditions fixed past 96 hours.

*2.3.2.4 Ocean Model Bathymetry and Vertical Grid Setup (***setup_nl_setupl***)*

Four parameters must be changed in **setup_nl_setupl** (Appendix B-3) by the user to customize the NCOM bathymetry and vertical grid levels.

- *lo* – This parameter specifies the number of vertical levels used in NCOM. The default is 51. To change *lo*, open the **setup_nl_setupl** script and change it to the desired number of levels, e.g., *lo* = 50, 50, 50… The *lo* parameter is set by *kkom* and *kkosm* in **gridnl.ocean**.
- *lso* - Specifies the number of sigma levels, with a default of 36 levels. This may also be altered to the user's specifications, e.g., *lso* = 35, 35, 35, etc. The *lso* parameter is set by *kkom* and *kkosm* in **gridnl.ocean**.
- *llog* - Changes the starting level for vertical grid stretching. The default is 1, which stretches the entire grid. When *llog* = 8, for example, the stretching begins at vertical level 8.
- *dztop* - The thickness of the surface layer for each layer until the *llog* level. Therefore, if *llog* = 8, the model will have seven layers above it of constant thickness (with *dztop* = 0.5). The default value is 1.0.
- *dmin* – The minimum depth on the model grid (in m, positive up). The default value is -2.0.
- *dmax* – This is the maximum depth on the model grid (in m, positive up). The default value is -1261.6873.
- *strfac* – This is the log stretching factor for the vertical grid. Its default value is 1.153018475.
- *dmaxout*- The maximum depth (in m, positive up) for z-levels. The default value is -3000.0. It is written to the OZOUT file.

These parameters are important for the SWAN grid setup. SWAN actually uses the NCOM setup for part of its grid setup.

- *lwav*- The number of vertical depths for wave forcing input. Default value is 50.
- *dmax_wav* - The maximum depth of the wave forcing input vertical grid (in m, positive up). The default value is -300.0.
- *dztop_wav* - The thickness (in m) of the top layer of the wave forcing vertical grid. The default value is 0.1.

The bathymetry, river, and tide files used by NCOM are also specified in this namelist. The default bathymetry database is the Digital Bathymetric Data Base- Variable resolution (DBDBV) and the file currently used is **dbdb2_v30.dat.**

*2.3.2.5 NCODA Namelist Parameter Setup (***setup_nl_oanl***)*

The NCODA program is set up and run primarily through the manipulation of namelist parameter files. The **setup_nl_oanl** script contains ocean analysis parameters specific to optimizing NCODA MVOI or 3DVar data assimilation for a COAMPS run. Table 9 below does not include all **oanl** namelist parameters available to NCODA. The default values listed here are the suggested settings for COAMPS and do not necessarily reflect original NCODA default values. The parameters here are for both NCODA MVOI and NCODA 3DVar. For a complete description of NCODA and its **oanl** namelist parameters, see Cummings and Carroll, 2006.

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)

Name	Type	Description	Default Values
argo_bias	logical	(true) Performs bias correction (drift adjustment) of Argo float salinities, which can drift over time due to bio-fouling, changes in calibration, etc. It is based on the GDEM salinity climatology.	default = .true.
bv_chk	real	Brunt-Vaisala frequency (cph) threshold.	default = 2.0

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)			
Name	**Type**	**Description**	**Default Values**
debug	logical	(true) Generates formatted output files to diagnose problems or monitor results from the analysis. 1) Argo salt bias correction, pooling of profile moorings and profile rejections (fort.32), 2) Profile inflexion point, standard level data, and vertical extension results using background fields (fort.33), 3) Geopotential profile observations (fort.34), 4) Observation and prediction errors (fort.35), 5) Listing of MVOI observations (fort.36), 6) Synthetics (direct and MODAS) for MODAS, including rejections and editing results (fort.31), and 7) Layer pressure observations (fort.38), 8) Produces diagnostics from 3DVar analysis such as processor assignment, volume interaction scales, etc., 9) Produces diagnostics about ensemble transformation file Input/Output (I/O; stdout) in the NCODA_ET program, or 10) Retains temporary volume solution files generated in the analysis step when the *himem* option is set true after post processing by the NCODA_POST program.	default =.true.
direct	logical	(true) Altimeter SSH anomaly data are assimilated by the Cooper-Haines (CH) method if *direct* is set true. The CH method is preferred if NCODA v3.43 is cycling with a skillful forecast model. The method cannot explicitly correct for model errors and so should not be used when the analysis is cycling on itself or the forecast model has systematic errors or biases.	default = .false.
diurnal	logical	(true) Daytime satellite SST retrievals are biased warm when the insolation is high and wind speeds are low. The NCODA_QC system flags satellite SST retrievals in diurnal warming events. Set *diurnal* to "true" to use these flagged observations in the NCODA v3.0 analysis.	default = .true.

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)			
Name	**Type**	**Description**	**Default Values**
fcst	logical	(true) Uses forecast model backgrounds on input. It is analysis variable dependent: 1) Ice, 2) SST, 3) SSH, 4) multivariate, 5) SWH 6) velocity.	default = .true.
fgat	integer	First guess appropriate time update interval (hrs) for: (1) Ice (%), (2) SST (deg C), (3) SSH (dynamic m), (4) Multivariate (dynamic m), (5) SWH (m) or (6) Velocity.	default = -1 (for all)
hc_mdl	char	Controls how the horizontal correlation length scales are defined in NCODA v3.0. If *hc_mdl* is set to 'locl', then NCODA v3.0 will attempt to read a 2D array of correlation length scales from a local file. It is anticipated that 'locl' will be used when the horizontal correlations are computed from an ensemble run of the analysis/forecast system. Horizontal correlation model options: 'rsby' = Rossby radius deformation, 'homo' = homogeneous scales, and 'locl' = user defined scales for the analysis variables: (1) Ice (%), (2) SST (deg C), (3) SSH (dynamic m), (4) Multivariate (dynamic m), or (5) SWH (m).	default = 'rsby '(for all)
himem	logical	(true) Executes analysis using I/O of analysis volumes rather than *mpi_reduce.*	default = .true.
locn3d	logical	(true) Performs a 3D analysis on this grid nest.	default = 7*.false.
lvl_nmo	real	Level of no motion for geopotential calculations. The level of no motion varies with the ocean basins. It is shallower in the Pacific and deeper in the Atlantic.	default = 1000 m

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)			
Name	**Type**	**Description**	**Default Values**
mask_opt	char	Grid mask options ('0D' '2D' or '3D'): '0D'= water only, '2D' = extends water points into land and below bottom, or '3D' = land and below bottom points eliminated.	default = '3D'
modas	logical	(true) Performs assimilation of altimeter Sea Surface Height Anomalies (SSHA) using MODAS synthetic profiles. Salinity is computed from the synthetic temperatures using historical TS relationships that are also stored in the MODAS database.	default = true
model	char	Forecast (NCOM, HYCOM) or wave model (WAVEWATCH III) region name.	default = 'ncom'
n_pass	integer	Number of 3x3 smoother passes on full valued analysis output fields.	default = 2
offset	real	(i, j) offsets from the grid boundary to restrict data selection for the assimilation (only for nest 1): from north (grid top), from south (grid bottom), from east (grid right), or from west (grid left). When NCODA v3.43 is cycled with a forecast model that has open boundaries, the forecast fields in the boundary zones often reflect the LBCs more than the forecast. Accordingly, innovations computed from observations in these boundary zones are likely to be very inaccurate and may adversely affect the forecast away from the boundaries depending on the covariances used to spread the innovations to the surrounding grid points.	Default = 10., 10., 10., 10.,

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)			
Name	**Type**	**Description**	**Default Values**
pool	logical	(true) Pools satellite systems for: 1) DMSP F11, F13, F14, F15, F16; 2) NOAA 14, 15, 16, 17, 18 GAC SSTs; 3) TOPEX, ERS, GFO, JASON, ENVISAT; 4) GOES 8, 10, 11, 12 SSTs; 5) NOAA 16, 17, 18 LAC SSTs; 6) Altimeter / Buoy SWH; 7) AMSRE, AMSR, TRMM microwave SSTs; 8) ATSR, AATSR skin SSTs; 9) MSG day/night SSTs; 10) METOP A, B, C GAC SSTs; 11) METOP A, B, C LAC SSTs.	default = t, t, f, t, t, f, f, f, f, f, f
prf_slct	real	Profile selection criteria options for : 1) acceptable level probability gross error, 2) minimum number of sampling levels, 3) minimum ratio of last sampling depth and bottom depth, 4) minimum sampling depth (if profile has not sampled the water column), 5) maximum acceptable depth distance between adjacent levels, 6) maximum acceptable temperature difference between adjacent levels, 7) maximum acceptable depth difference at level of maximum temperature difference, (8) maximum acceptable temperature difference at level of maximum level difference, (9) maximum depth first sample, or (10) maximum surface extrapolation depth.	default = 20

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)			
Name	**Type**	**Description**	**Default Values**
prf_time	char	Profile time sampling options: 'cycl' = selects profiles based on the number of *prf_hrs* specified, 'obst' = selects profiles based on the time the profile was observed, 'rcpt' = selects profiles based on the time the profile was received at the center, and 'synt' = selects obs that are synoptic for the analysis update interval. For real-time analyses, the 'rcpt' option is useful to ensure that profiles received late are still selected. For delayed-mode analyses, the 'obst' option is useful to ensure that only profiles observed in the period of the analysis time window are chosen. For these two *prf_time* options, the look-back period for profile observations is a floating DTG determined by the youngest observation assimilated in the previous analysis. Profile data selection can also be forced to coincide with the analysis time window by setting *prf_time* to 'synt'.	default = 'obst'
pt_anl	logical	(true) Potential temperature analysis. Observations are reported as in situ temperatures, while ocean forecast models use potential temperature. If *pt_anl* is set true, NCODA v3.0 converts all in situ temperatures from observations and climate databases to potential temperature. The analyzed temperature increments become potential temperature increments and can be used directly to correct the model forecast temperatures in a sequential incremental update cycle.	default = true
rscl	real	Horizontal correlation length scales based on the Rossby radius of deformation can be adjusted by multiplying with a scalar value defined in *rscl* for a given analysis variable. The adjustment can be made to increase or decrease the length scales interpolated from the 1° resolution Rossby radius database to the analysis grid. The Rossby radius scaling factor is for: 1) ice, 2) SST, 3) SSH, 4) MVOI, and 5) SWH analysis variables.	default = 2.0, 1.2, 1.2, 1.2, 2.0

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)			
Name	**Type**	**Description**	**Default Values**
sal_asm	logical	(true) Assimilates real salinity observations.	default = .true.
ssh_asm	logical	(true) Performs altimeter SSH assimilation.	default = .true.
ssh_mean	char	When altimeter SSHA data are assimilated via synthetic profiles generated by the CH algorithm (namelist parameter *direct* is set true), NCODA v3.43 requires a SSH mean field. SSH mean field options: 'modl' = model mean field, or 'data' = data derived mean field.	default = 'data'
ssh_std	real	Max number of standard deviations to scale altimeter SSH from climatology. If an altimeter SSHA observation exceeds *ssh_std*, then it is scaled to fall within *ssh_std* + 1 standard deviation of the dynamic height variability that has been computed from 50 years of historical profile data and stored in the MODAS database.	default = 2.0
ssh_time	char	Altimeter SSH processing option: 'cntr' = select obs in data window centered around analysis time; 'cycl' = select obs based on full TOPEX repeat cycle (10 days); 'obst' = select obs based on time SSHA was observed; 'rcpt' = select obs based on time SSHA was received at the center; or 'synt' = select obs that are synoptic for analysis update interval.	default = 'obst'
sst_asm	logical	(true) Performs SST assimilation. Turning off the assimilation of SST data (*sst_asm* set false) may be useful in large domains when there are millions of SST observations. If *sst_asm* is set false, then the forecast model is typically relaxed to the SST corrections computed in the NCODA v3.43 2D analysis.	default = .true.
st_grd	logical	(true) Generates SST observations for 3D MVOI from the analyzed SST grid.	default = true (for all grid nests)
st_smpl	real	Analyzed SST observation sampling.	default = 8

Table 9: NCODA v3.0 Ocean Analysis Namelist (*oanl*)			
Name	**Type**	**Description**	**Default Values**
upd_cyc	integer	Analysis update cycle (hours). Background fields from a previous analysis or a model forecast must be available in the restart directory at the prescribed update cycle interval. Changing *upd_cyc* during a cycling analysis run may force NCODA v3.0 to cold start if restart files at the update cycle are missing.	default = ${update_cycle}
vc_bkg	char	Vertical correlation background: 'clim' = climatology; 'fcst' = model forecast.	default = 'fcst'
vc_mdl	char	Vertical correlation model options: 'mixl' = mixed layer, 'dens' = correlation length scales defined by first guess vertical density gradients, 'cons' = constant, or 'isop' = vertical correlations defined along isopycnal surfaces.	default = 'dens'
vol_scl	real	Factor to control the size of observation volumes. It scales the maximum horizontal length scale on the analysis grid and can increase (>1) or decrease (<1) volume sizes.	default = 6, 4, 6, 6, 8
z_lvl	real	Analysis vertical grid (m). **Note**: if *z_opt* is set to 'ncom', then *z_lvl* contains NCOM velocity levels.	default = ${z_lvl}
z_opt	char	Vertical grid options: 1) 'hycom' - read HYCOM vertical grid 2) 'ncom ' - compute sigma/*z* vertical grid 3) 'none ' - use vertical grid.	Default = ${z_opt}

2.3.2.6 ESMF Configuration (setup_esmf_config)

In the ***projects/$area*** directory, the **setup_esmf_config** script specifies the variables being passed between the models. It also specifies the coupling interval between the atmospheric and ocean models. A copy of the script is found in Appendix B-4.

Important variables:

- *cpl_sec* - Specifies the coupling interval for a COAMPS (in seconds) coupled forecast. The time interval must be divisible by the time step of both the atmospheric and ocean models. The default value is currently set at 360 seconds (6 min).

- *ocean_export_init_only* – Defines how ocean fields are exchanged. If set to true, then only initial ocean fields are exchanged (when *couple_o2a*=t). If set to "f" the time-varying ocean fields are exchanged (when *couple_o2a*=t). The default value is "f".
- *wave_export_chnk_type* – This variable specifies the type of Charnock drag coefficient scheme useful in tropical cyclone simulations. It is important for wave to atmosphere feedback but it is currently experimental only. Option 1 is the old scheme of Janssen (1991) (1=*w2a*_janssen) and option 2 is a new scheme developed by Moon et al. (2004) (2=*w2a*_moon).

There are several coupling variables passed from model to model (Table 10). These can be turned off by simply deleting the variable from the **setup_esmf_config** script within the user's ***/projects/area*** directory for the given COAMPS run.

Table 10: Coupling variables passed between models.

Variable	From/to	Description
pmsl	atmos to ocn	Air pressure at sea level.
tauu	atmos to ocn	Surface downward eastward stress.
tauv	atmos to ocn	Surface downward northward stress.
hfns	atmos to ocn	Surface downward heat flux.
mfns	atmos to ocn	Surface downward moisture flux.
risw	atmos to ocn	Isotropic shortwave radiance in the air.
sst	ocn to atmos	Sea surface temperature.
wndu	atmos to wave	Eastward 10 m wind.
wndv	atmos to wave	Northward 10 m wind.
chnk	wave to atmos	Wave induced Charnock parameter.
wvst		Surface total wave induced stress.
wvsu		Surface eastward wave induced stress.
wvsv		Surface northward wave induced stress.
ssh	ocn to wave	Sea surface height above sea level.
sscu	ocn to wave	Surface eastward sea water velocity.
sscv	ocn to wave	Surface northward sea water velocity.
sdcu	wave to ocn	Eastward Stokes drift current.
sdcv	wave to ocn	Northward Stokes drift current.
wbcu	wave to ocn	Eastward wave bottom current.

Variable	From/to	Description
wbcv	wave to ocn	Northward wave bottom current.
wbcf	wave to ocn	Wave bottom current radian frequency.
wsuu		Eastward wave radiation stress.
wsuv		Eastward-northward wave radiation stress.
wsvv		Northward radiation stress.
wsgu	wave to ocn	Eastward wave radiation stress gradient.
wsgv	wave to ocn	Northward wave radiation stress gradient.

```
cpl_atmos_to_ocean: ${couple_a2o}
cpl_atmos_to_ocean_field_list: pmsl tauu tauv hfns mfns
cpl_ocean_to_atmos: ${couple_o2a}
cpl_ocean_to_atmos_field_list: sst
cpl_atmos_to_wave: ${couple_a2w}
cpl_atmos_to_wave_field_list: wndu wndv
cpl_wave_to_atmos: ${couple_w2a}
cpl_wave_to_atmos_field_list: chnk
cpl_ocean_to_wave: ${couple_o2w}
cpl_ocean_to_wave_field_list: ssh sscu sscv
cpl_wave_to_ocean: ${couple_w2o}
cpl_wave_to_ocean_field_list: sdcu sdcv wbcu wbcv wbcf wsgu
wsgv
```

2.3.2.7 Post Processing Output File Setup (setup_nl_postnl)

COAMPS now will automatically produce netCDF file output given NCOM output flatfiles (**setup_nl_omnloff**). When specifying netCDF output intervals, please be sure the ocean flatfile output is being generated (at least) at the interval established in **setup_nl_omnloff**.

There are 12 namelist parameters within **setup_nl_postnl** (See Appendix B-13) that dictate the characteristics of the flatfiles being output. Two important parameters are:

- *post_tauinc*- The frequency of NCOM output to process (hours). It must be a multiple of the required off* flatfile output intervals. The default value is 3 hours.
- *post_taufile* – If set to "t", an individual netCDF output file for each tau will be generated. The default value is 't'.

2.3.3 SWAN Wave Model Setup (setup_swan_input)

SWAN is command driven and no manipulation of namelist parameters is required. To run SWAN within COAMPS the **setup_swan_input** (Appendix B-14) file creates input data necessary for the coupling.

2.3.3.1 Setting the SWAN Timestep

dtw- is the timestep for SWAN in seconds. The default is 60.

2.3.3.2 Configuring the SWAN Computational Grid

The SWAN wave grid setup utilizes the NCOM grid setup program for grid creation. Nesting in SWAN is not currently available in COAMPS; however, a SWAN grid can be setup to be exactly the same as the ocean grid (with respect to size and resolution) or with a differing resolution grid.

To create a SWAN grid that is exactly the same as the model NCOM grid, perform the following steps:

1. Copy the **gridnl.ocean** namelist to the **gridnl.wave** namelist.
2. Remove the *kkom* and *kkosm* parameters from **gridnl.wave**.
3. Run the wave setup by including the *-w1* option in the run command line.
4. A **bottom.dat** file (bathymetry file for SWAN based on DBDB2 bathymetry) and **grid.dat** file (horizontal grid) will be created for SWAN. The **ohgrd** files will be produced as well since the NCOM setup is used. These files are located in the ***$outDataDir/wave*** directory.

A SWAN grid of differing size and resolution may also be created by first generating a new **gridnl** and placing the grid information into **gridnl.wave**. Follow the same steps for creating a new **gridnl** in COAMPS-OS, outlined in Section 2.4. Please make sure the SWAN grid is inside the course atmospheric grid.

For changing wave frequencies and directions, locate this following section of code within the **setup_swan_input** script in Appendix B-14) by clicking on the hyperlinked section below:

CGRID CURV ${mcx} ${mcy} EXCEPTION 999 999 CIRCLE 36 0.0418 1.0 33.

The number 36 in this section of code is the number of wave directions output by the wave model. Thirty-six directions correspond to 360 degrees (CIRCLE 36 = 10 degree intervals). The number 0.0418 is the highest wave frequency in SWAN (Hz), and the number 33 represents the number of frequencies output by the wave model.

For instructions more specific to setting up and running SWAN, see the SWAN Technical Document (2010) at http://iod.ucsd.edu/~falk/modeling/swantech.pdf. A validation study of SWAN was conducted by Allard et al in 2004.

2.3.3.3 Setting Wave Source Terms

There are two types of wave source term physics available in SWAN. Komen (1994) physics is the older, standard SWAN physics. Babanin physics is new to SWAN (Babanin et al. 2010, Rogers et al. 2011), is observation based, and shows vast improvement in high-wind regimes such as tropical cyclones. Both Komen and Babanin physics may be used in lower wind regimes. From the portion of script below, simply uncomment the physics scheme desired.

```
$ Source/sink terms and conv. criterion
$ *** Old Physics ***
$ GEN3 KOMEN AGROW
$ WCAP KOM 2.36E-5 3.02E-3 2.0 1.0 1.0
$ *** New Physics ***
GEN3 BABANIN 5.7E-7 8.0E-6 4.0 4.0 1.2 0.0060 UP AGROW
```

2.3.3.4 Setting the Wave Propagation Stability Parameter

PROP GSE 12 HR - This is the wave propagation "garden sprinkler effect" stability parameter. Higher resolution wave grids will generally need a lower GSE time to aid in stability and to lower these effects. Comment out this line for a wave grid resolution of less than 500 m.

2.3.3.3 Establishing Output Requests

Output requests may be in MATLAB, 1D or 2D Spectra, or ASCII text file formats.

In the following excerpt from the **setup_swan_input** script, the MATLAB output option is uncommented in order to generate that output format. To generate ASCII or 1D/2D spectra formats, simply uncomment those portions of the script.

```
$
$ *** MATLAB Output ***
BLOCK 'COMPGRID' NOHEAD 'grid_${ddtg}.mat' LAY 3 &
  DEPTH XP YP OUTPUT ${start_day} 100 DAY
BLOCK 'COMPGRID' NOHEAD 'output_${ddtg}.mat' LAY 3 &
  ${output_fields} &
  OUTPUT ${start_day} 1 HR
$
$ *** ASCII Output ***
$ BLOCK 'COMPGRID' HEAD 'grid_${ddtg}' LAY 4 &
$   DEPTH XP YP OUTPUT ${start_day} 100 DAY
$ BLOCK 'COMPGRID' HEAD 'output_${ddtg}' LAY 4 &
$   ${output_fields} &
$   OUTPUT ${start_day} 1 HR
$
$ *** 1D Spectra Output ***
$ SPECOUT 'COMPGRID' SPEC1D ABSOLUTE 'spec1d_${ddtg}' &
```

```
$   OUTPUT ${start_day} 1 HR
$
$ *** 2D Spectra Output ***
$ SPECOUT 'COMPGRID' SPEC2D ABSOLUTE 'spec2d_${ddtg}' &
$   OUTPUT ${start_day} 3 HR
$
$ *** Nest Output ***
$ NONE
```

Point output may also be specified. The following is an example of command line that can be placed in the OUTPUT section for point output:

```
$ POINTS '42036' 275.489722 28.506111
$ TABLE '42036' NOHEAD 'table_42036_${ddtg}' &
$   TIME DEPTH XP YP ${output_fields} &
$   OUTPUT ${output_sday} 15 MIN
$ SPEC '42036' SPEC2D 'spec2d_42036_${ddtg}' OUTPUT
${output_sday} 15 MIN
```

In this example `'42036'` is a label, `275.489722` is the longitude for the point, `28.506111` is the latitude for the point, `15 MIN` is the time interval of output, `SPEC2D` is 2D spectra output, and `'spec2d_42036_${ddtg}'` is the file name that will contain the 2D spectra for the point. `TABLE` displays ASCII wave output for the point and `'table_42036_${ddtg}'` is the name of the file that will contain the ASCII output.

2.3.4 WAVEWATCH III Model Setup

Full instructions for running WAVEWATCH III may be found at
http://polar.ncep.noaa.gov/mmab/papers/tn276/MMAB_276.pdf

The **setup_ww3_strt** script (Appendix B-15) provides the initial conditions to the WW3 model. The preprocessing input file script, **setup_ww3_grid** (Appendix B-16), defines the WW3 grid. The **setup_ww3_multi** (Appendix B-17) script sets up the multi-grid, or mosaic grid, model driver input. WW3 grid output post processing is accomplished through the **setup_ww3_outf** (Appendix B-18) script. Point output post processing is done through the **setup_ww3_outp** (Appendix B-19).

2.4 Atmospheric Nest Setup Using COAMPS-OS®

The fastest and easiest method to set up the COAMPS atmospheric grids is to use COAMPS-OS® (COAMPS ON-SCENE). In addition to the COAMPS modeling components, COAMPS-OS has web-based interfaces to configure COAMPS, access data, and automate graphical processing. It also includes software that interacts with the Navy's Tactical Environmental Database Services (TEDS) (Geiszler et al., 2003).

A username and password are required to login to the COAMPS-OS system, which can be obtained by contacting either Travis Smith (travis.smith@nrlssc.navy.mil; 228-688-5631) or David May (david.may@nrlssc.navy.mil; 228-688-4707), who are both employed at NRL. At NRL-SSC, access to COAMPS-OS is available on *FST1* at http://FST1/COAMPS-bin/COAMPSOS_homepage.cgi (See Figure 1). Follow the steps below to produce the atmospheric **gridnl** namelist to be transferred to the **gridnl_atmos** namelist in the ***/projects/$area*** directory.

Figure 1: COAMPS-OS Homepage on *FST1*.

2.4.1 Defining the Grids

After logging on to COAMPS-OS, click the general area on the world map where the grids will be setup (See Figure 2). Two grids will appear in the area that was chosen, an outer grid (white) and an inner grid (yellow). To zoom into the area, click and drag a box around the area of interest and then click the **Zoom In** button below the map.

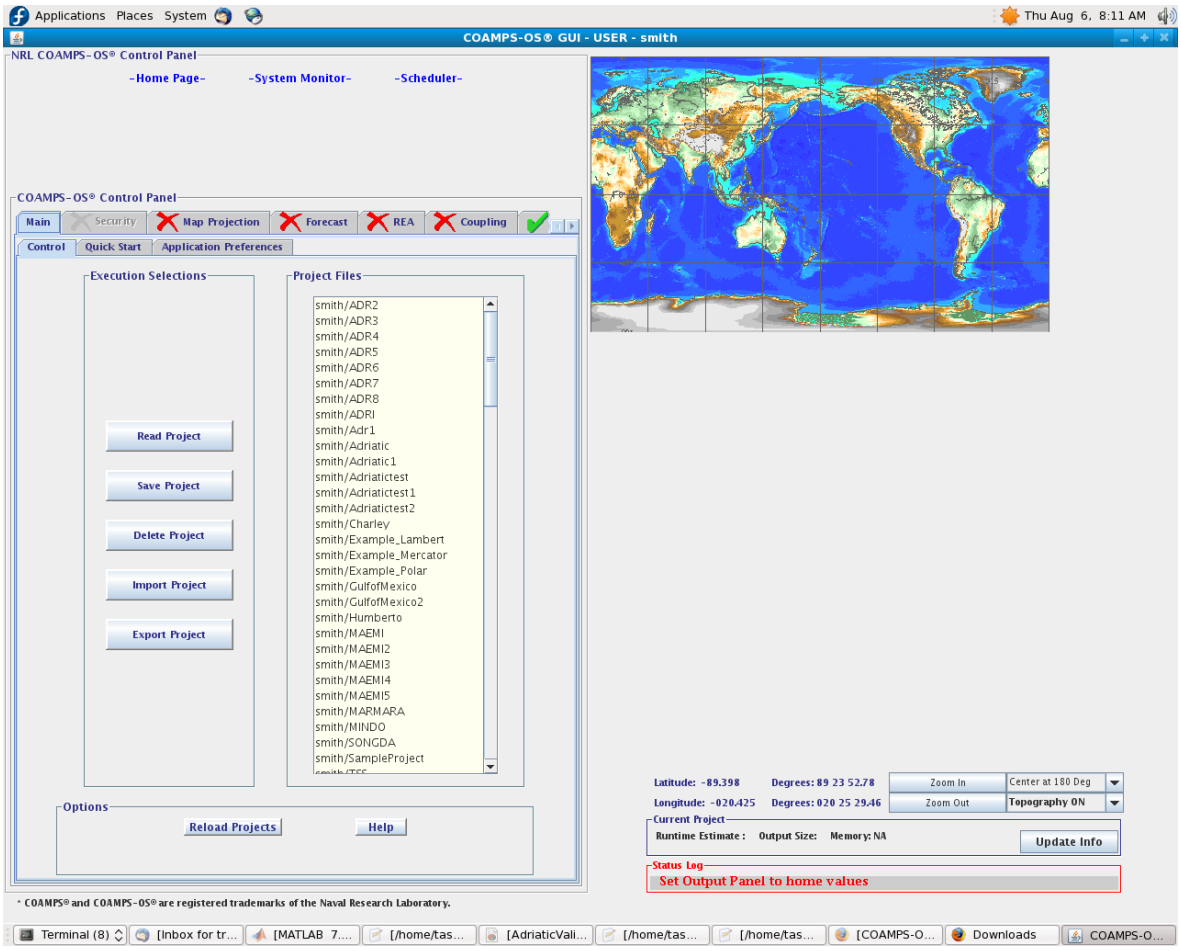

Figure 2: COAMPS-OS Main GUI Page.

2.4.2 Configuring the Nests

To adjust the size, location, projection, and number of nests, click on the **Map Projection** tab in the COAMPS-OS Control Panel. There are three tabs within the Map Projection section: **Location**, **Positioning**, and **Options**.

2.4.2.1 Choosing a Map Projection

Click on the **Location** Tab (See Figure 3). Choose the desired map projection (Mercator, Lambert Conformal, Polar Stereographic, or Spherical). Lambert Conformal may only be used between 20 and 70 degrees of latitude. If a nest is partially located south of 20°N (Northern Hemisphere), the Mercator projection will automatically be chosen.

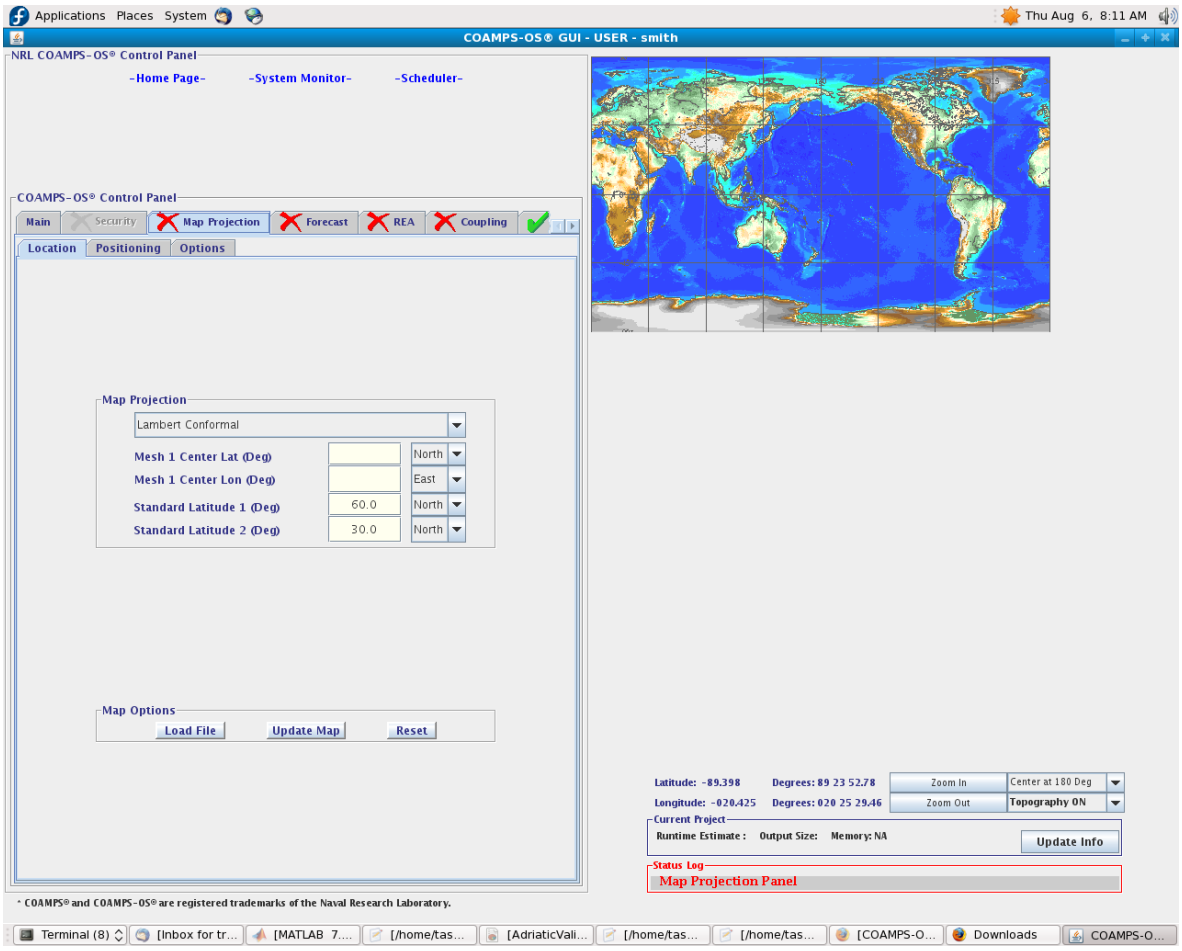

Figure 3: COAMPS-OS Map Projection tab showing the Location tab options.

2.4.2.2 Optimizing Nests

The **Positioning** tab is used to adjust the number, resolution, size, and position of each nest (See Figure 4).

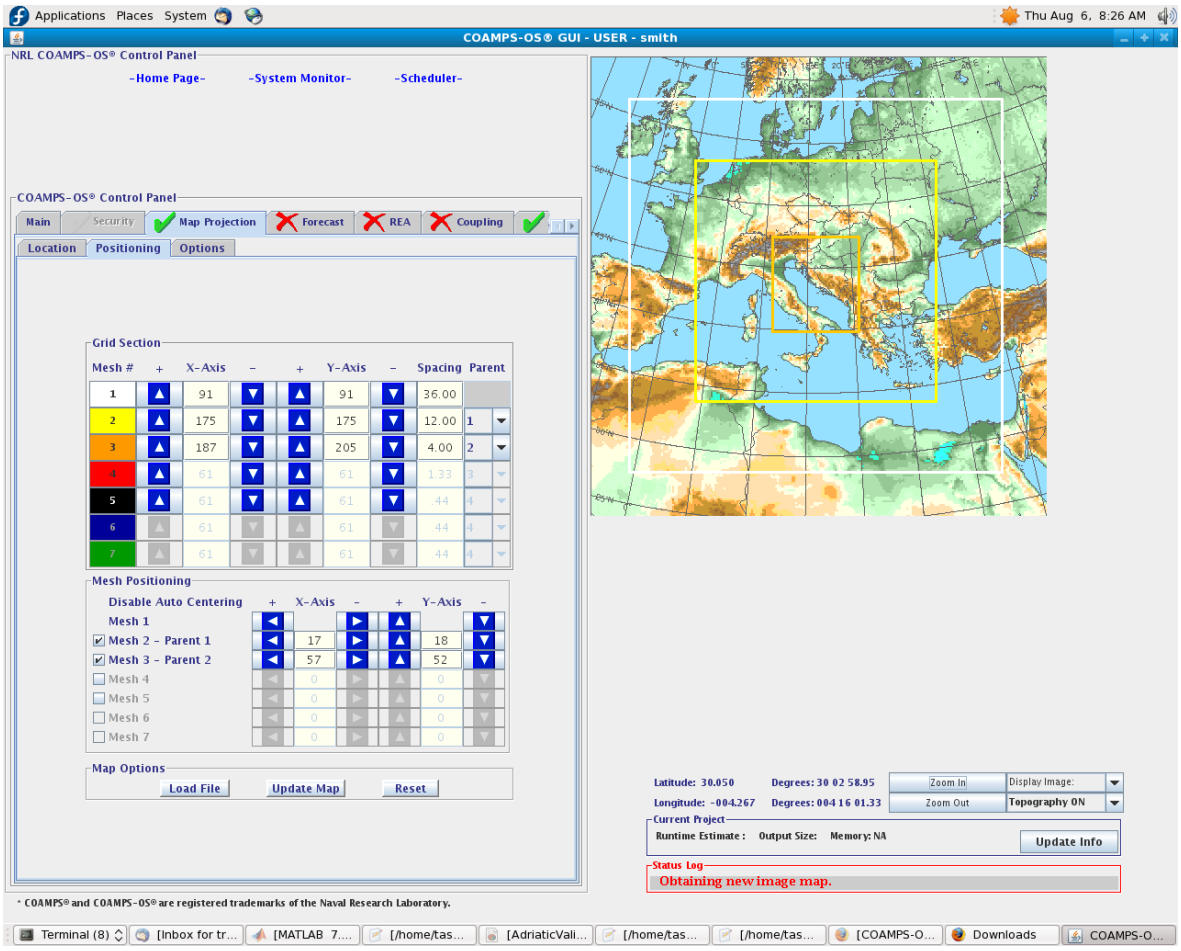

Figure 4: Positioning tab view of the Map Projection window.

a. Number of Nests: To choose three (or more) nests, click on the "3" in the **Mesh #** column. An orange nest will appear on the map for nest 3.

b. Nest Resolution: Adjust the resolution of the nests in the spacing column. The default for three nests is 54, 18, and 6 km. If a change in nest resolution is made, COAMPS-OS will automatically adjust the spacing of the other nests to a 3:1 ratio. For instance, if the spacing of nest 1 is changed to 27 km, the spacing for nest 2 and nest 3 will automatically change to 9 and 3 km, respectively.

c. Grid Size: Adjust the grid size by clicking the up and down arrows in the X-axis and Y-axis columns. The number of grid points of each nest is displayed.

d. Grid Position: Adjust the position of each of the grids in the **Mesh Positioning** section. There are two ways nest positioning can be accomplished. To keep the inner nests centered with respect to the outer nests, just move Mesh 1 by clicking on the up, down, left, and right arrows in the X-axis and Y-axis columns. To move an individual nest within the primary grid, click the boxes next to the **Mesh #** and then use the arrows to reorient it.

2.4.3 Saving the COAMPS-OS Project

After the nest setup is complete, click on the **Run** tab in the COAMPS-OS Control Panel (Figure 5). A prompt will appear to save the current project. Click **Cancel**. A prompt will then appear to save the project. Click **OK**. The next prompt will ask about run confirmation. Click **OK** and then enter a name for the project in the **Save File As** field then click **OK** again.

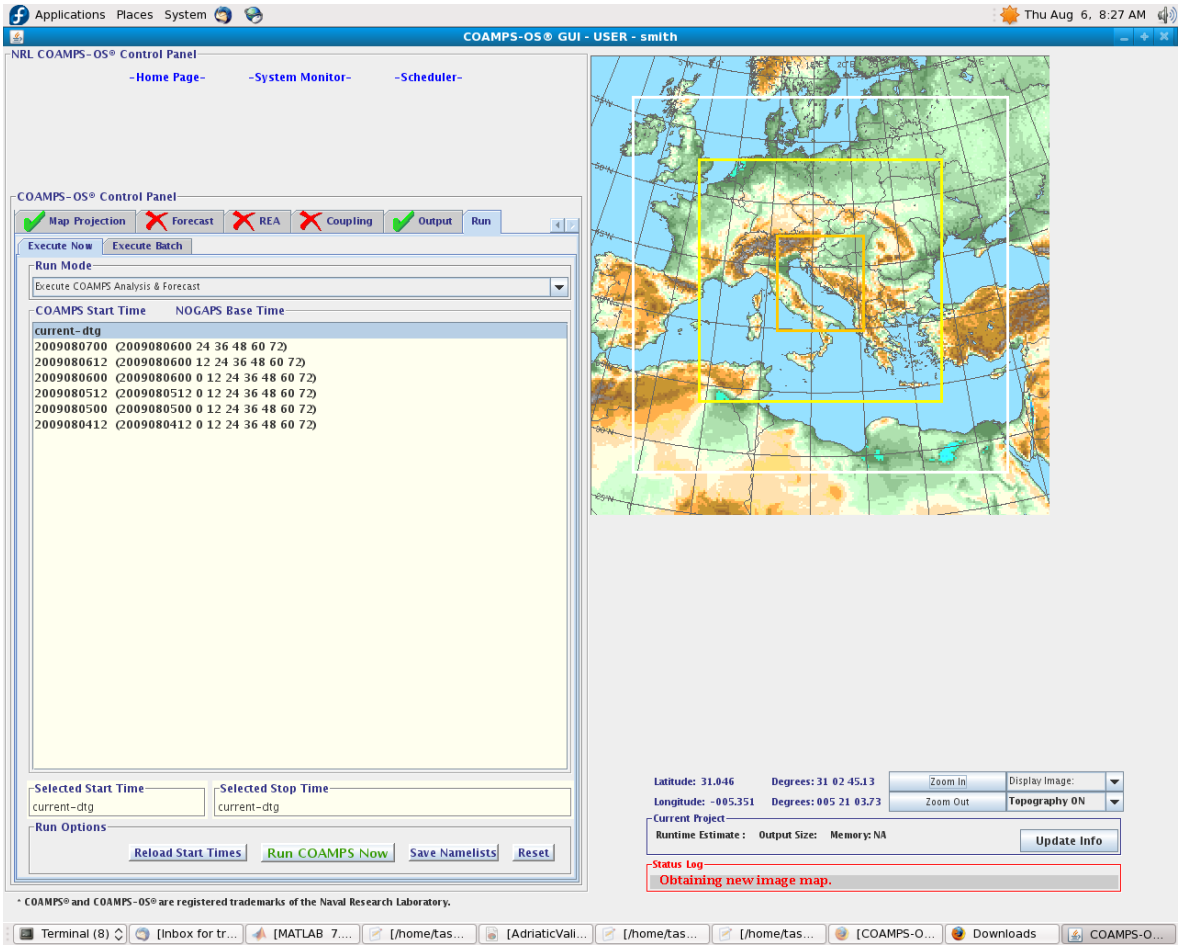

Figure 5: The Run tab on the Control Panel offers several options for setting up and executing a COAMPS-OS run.

2.4.4 Saving the COAMPS-OS Namelists

After the project is saved, click on the **Run** tab again in the COAMPS-OS Control Panel. Click **OK** at the prompt. Click on the **Save Namelists** button under the **Execute Now** tab. The namelist **current-dtg.nl**, which contains the nest information, will be saved to a location that the user specifies.

2.4.5 Transferring Grid Information to COAMPS

Find the **[name of project].current-dtg.nl** file, saved to a user-specified location (Section 2.4.4). The grid information is under the ***&gridnl*** title heading at the top of the file. The **gridnl** information must be *exactly copied* into the **gridnl_atmos** namelist in the ***/projects/$area*** directory.

3.0 RUNNING A SIMULATION

3.1 Execution of the Run Script

The functionality of the **run_coamps** script in the ***jobs/$area*** directory includes a number of arguments that may be specified for a specific case. These include simulations that are fully coupled, one-way, or stand alone as well as runs with or without data assimilation. The run script must be executed in the ***jobs/$area*** directory, and the following command, including arguments, must be specified:

```
$RUNSCRIPTNAME [options] [beginning date-time-group],
```

where $RUNSCRIPTNAME is the name of the run script including its location relative to the ***jobs/$area*** directory.

3.1.1 Run Script Options

The following sections offer several choices to customize the user's COAMPS run. Examples of run script commands are found in Section 3.2.

3.1.1.1 Observational Data Options (-d)

-d Command Choice #	Description
0	No action. (This is the default if all data are already in the appropriate directories).
1	Get the ADP (atmospheric) data archived on *NEWTON*. They are available for COAMPS runs at the DSRC (e.g., *DAVINCI or EINSTEIN*) only.
2	Get the OCNQC (ocean quality controlled) data on *NEWTON*. They are available for COAMPS runs at the DSRC (e.g., *DAVINCI or EINSTEIN*) only.
3	Get the ADP and OCNQC data.

3.1.1.2 Global Model Data Options (-g)

-g Command Choice #	Description
0	No action (default).
1	Get the NOGAPS and ADP fields archived on *NEWTON*. They are available for COAMPS runs at the DSRC (e.g., *DAVINCI or EINSTEIN*) only.
2	Get the global NCOM fields archived on *NEWTON*. They are available for COAMPS runs at the DSRC (e.g., *DAVINCI or EINSTEIN*) only.
3	(1) and (2).

3.1.1.3 Atmospheric Analysis Options (-a)

-a Command Choice #	Description
0	No action (default).
1	Run the atmospheric analysis.
2	Run the ocean analysis (NCODA) on atmospheric grids.
3	(1) and (2).

3.1.1.4 Ocean Analysis Options (-o)

-o Command Choice #	Description
0	No action (default).
1	Run the ocean setup.
2	Run the ocean analysis (3D NCODA) on ocean grids.
3	(1) and (2).

3.1.1.5 Wave Analysis Options (-w)

-w Command Choice #	Description
0	No action (default).

1	Run wave setup.

3.1.1.6 Forecast Options (-f)

-f Command Choice #	Description
0	No action (default).
1	Run coupled forecast.
2	Run stand-alone atmospheric forecast.
3	Run stand-alone ocean forecast.
4	Run stand-alone wave forecast.

3.1.1.7 Post-processing Options (-p)

-p Command Choice #	Description
0	No action (default).
1	Run ocean post processing.
2	Run wave post-processing.
3	(1) and (2).

3.1.1.8 Ending Date-Time-Group Option (-e)

-e Command Choice	Description
-e	yyyymmddhh (defaults to starting DTG).

3.2 Run Command Examples

3.2.1 Fully Coupled Air/Sea/Wave Run with Atmosphere and Ocean Assimilation

For a fully coupled air/sea/wave run with both atmosphere and ocean data assimilation (including all the setup commands for each of the individual model components), the following run command should be executed in the ***jobs/$area*** directory. Option `-a1` must always be specified to run the atmospheric assimilation (i.e., the atmospheric assimilation in COAMPS cannot be switched off). The following command is for running COAMPS *only* if all the input data are available:

`../../scripts/run_coamps -a1 -o3 -w1 -f1 -e yyyymmddhh` (ending DTG) `yyyymmddhh` (starting DTG)

The order of operations below must be followed when running COAMPS and can be found in the **log.[DTG]** file in the ***runDir/[DTG]*** directory:

1. Copy the **.ngt** files from the ADP directory to the ***runDir*** (**.ngt** files are used for tropical cyclone modeling).
2. Run the atmospheric analysis for initial and boundary conditions (MVOI or NAVDAS (highly recommended) as specified in **setup_nl_coamnl**).
3. Run NAVDAS (or 2D NCODA if MVOI is specified).
4. Run the ocean model setup.
5. Create ocean initial conditions.
6. Create ocean boundary conditions.
7. Create ocean background tendencies for SST outside of the NCOM domain.
8. Run 3D NCODA if ocean assimilation is turned on (option `-o3`).
9. Run the wave model setup.
10. Run the coupled forecast.

3.2.2 Two-way, Fully Coupled Run with Data Assimilation on DSRC Platforms

Use the following run command for execution on *DAVINCI or EINSTEIN* platforms.

`../../scripts/run_coamps -a1 -o3 -w1 -f1 -e yyyymmddhh` (ending DTG) `yyyymmddhh` (starting DTG).

3.3 Log Files

The log and command files are located in the ***outDataDir/$area/run/[DTG]*** directory for the user's specific DTG. Each DTG directory contains several log files. Table 11 describes the important log files.

Table 11: Log file descriptions.

Log File	Description
log.aanalysis	Atmospheric analysis log.
log.osetup.config	Ocean analysis log.
log.oanalysis.atmos	NCODA log.
log.cforecast	Coupled model log.
log.osetup.doinit	Creation of NCOM initial conditions log.
log.osetup.doobcs	Creation of NCOM boundary conditions log.
log.osetup.dohten	Creation of ocean background tendencies for SST outside of the NCOM domain.
log.wsetup.doswan	SWAN model setup log.
log.oanalysis.ocean	3D NCODA for ocean assimilation log.

3.3.1 Examining Model Run Times

Following a COAMPS run, the user may wish to look at COAMPS run times, either for the individual model components or for the entire run. In the ***run/[DTG]*** directory there will be **PET[#].ESMF_LogFile** files, with one for each processor employed in the simulation. Open the **PET0.ESMF_LogFile**. The start and end times flank the file text. Just above the final time stamp are several columns of summary data, including the run times of the individual model components, as shown below:

```
20110209 074802.758953   PET0 WTIME:     timer init_cnt    init_time    run_cnt     run_time final_cnt   final_time
20110209 074802.758978   PET0 WTIME:     TOTAL 1   0.178264E+05     1    0.106592E+05     1 0.464331E+03
20110209 074802.759031   PET0 WTIME:     ATMOS 1   0.380964E+02   120 0.162532E+04     1 0.211325E+02
20110209 074802.759087   PET0 WTIME:     WAVE  2   0.150977E+02   120 0.340487E+04     1 0.439596E+03
20110209 074802.759114   PET0 WTIME:     OCEAN 2   0.202843E+03   120 0.555854E+04     1 0.202330E+01
20110209 074802.759136   PET0 WTIME:     WBKGD 1   0.148392E-02   120  0.644064E-02    1 0.417111E-01
20110209 074802.759146   PET0 WTIME:     OBKGD  1  0.317280E-01   120 0.627309E+00     1 0.475719E-01
```

The **PET0.ESMF_LogFile** information should allow the user to determine a "sweet spot" with which to run COAMPS, based on run times and the number of processors designated to each model in the **setup_area** script (see Section 2.2.2.6).

3.4 Atmosphere, Ocean, and Wave Output Files

The output for both the atmosphere and ocean is a standard COAMPS binary flat file, as specified in the COAMPS User's Manual (Chen et al., 2003). If running COAMPS locally or on the Grid Engine, output for both the atmosphere and ocean is located in the output directories specified in the **setup_nrlssc** script. Output for COAMPS running on the DSRC (as specified in **setup_navy_dsrc**) is located in the ***/scr/[user]/COAMPS/data/$area*** directory. Native NCOM output can be specified in **setup_nl_oparm** in the ***projects/$area*** directory and will be output into the ***$outdataDir/ocean/[DTG]*** directory. Wave output from SWAN in MATLAB format (.mat) will be sent to the ***$outdataDir/wave/*** directory.

3.5 Running COAMPS-TC

COAMPS-TC is a modeling option in COAMPS that allows for realistic representations of tropical cyclones. It includes 3D variational analysis, synthetic observations for vortex initialization, improved microphysics, air-sea fluxes, and boundary layer processes for predicting tropical cyclone track, structure, intensity, and ocean response. The NAVDAS 3DVar atmospheric assimilation system contains a tropical cyclone bogusing technique to ingest wind, wind radii, and pressure data provided by the National Hurricane Center and Joint Typhoon Warning Center for an existing tropical cyclone in the Atlantic, Eastern, or Western Pacific basins. When using the COAMPS-TC build, COAMPS-TC generates a copy of the **.ngt[DTG]** from the ADP directory and sends it to the ***run/[DTG]*** directory to be used by COAMPS-TC. In order to use COAMPS-TC, the following options must be changed in the setup scripts:

1. In the **setup_area** script in the ***jobs/[PROJECT NAME]*** directory, make sure that the number of processors for NAVDAS, *atmosa_nprocs*, is four or greater, since only NAVDAS may be used for COAMPS-TC.

2. In the ***projects/[PROJECT NAME]*** directory, the following script parameters must be altered:

    a) ***gridnl.atmos***: COAMPS-TC uses moving nests that follow the tropical cyclone vortex center with each atmospheric time step. COAMPS-TC utilizes three atmospheric nests, with the two inner nests moving with the cyclone center. It is important when modeling a tropical cyclone that the coarse nest covers a large enough area to allow the two inner nests to translate within the coarse nest. The inner nests cannot translate or travel outside of the coarse atmosphere nests or the model will crash. Follow the normal setup for the atmospheric grids using COAMPS-OS. Switch on the moving nest options for nest 2 and 3 in the parameter *lnmove*. Set *lnmove* = 'f', 't', 't'. When using Lambert Conformal projection, make sure that *phnt2* is set to the same value as *phnt1* (*phnt2* will be set to -99.0 when using COAMPS-OS grid setup and Lambert Conformal projection).

    b) ***gridnl.ocean*** and ***gridnl.wave***: The ocean and wave grids must be large enough to accommodate the moving atmospheric nests or the model will crash (Appendices E and F).

    c) ***setup_nl_coamnl***: In the **coamnl** namelist, there are several options for tropical cyclone modeling. Since COAMPS-TC must run with NAVDAS, set *atmos_analysis_type* = 'navdas'. To turn on the tropical cyclone tracking, switch *ltctrk* = t. The parameter *ntcmov* specifies the nest that will track the tropical cyclone, so leave *ntcmov* = 3. The *tcid* parameter is the tropical cyclone identification number that identifies the number and location of the tropical cyclone. This information is found in the **ngt[DTG]** file, which contains the wind, wind radii, and pressure data that are used by NAVDAS to create the tropical cyclone vortex. Locate the **ngt[DTG]** file in the ***adp[DTG]*** directory, view it, and find the fourth and fifth columns. The fourth column is the tropical cyclone identification number and the fifth column is the location of the tropical cyclone. The location is delineated by the following letters:

        L: Atlantic basin
        E: Eastern Pacific basin
        W:Western Pacific basin

    For example, to model Hurricane Ida from 2009110800, the *tcid* = '11L', with 11 being the identification number (11th depression of the 2009 Atlantic hurricane season) and L for Atlantic from the

ngt2009110800 file. When starting COAMPS-TC, the **ngt[DTG]** file will be copied from the ***adp[DTG]*** directory to the ***runDir*** for use by NAVDAS to create the tropical cyclone vortex. The parameter *movemx* is the maximum number of grid points per iteration that a nest can move. Please don't alter *movemx*. The atmospheric time step *delta* is set to 120 and is the optimal value for using the moving nests. A different atmospheric timestep may be tested, but a model crash may occur.

3. The NAVDAS atmospheric analysis must be employed when using COAMPS-TC. NAVDAS will assimilate the data in the ***adp*** directory and use NOGAPS fields for the background. Generally, the current DTG NOGAPS fields (***f[DTG]*** directories) are required for NAVDAS, but NAVDAS can also look at the two previous NOGAPS DTGs (00Z and 12Z) for background as well. If running a hindcast or real-time simulation and beginning the simulation from the starting DTG, it would be helpful to grab the previous DTG NOGAPS fields as well, but NAVDAS will still run if these are not available.

4. To run COAMPS-TC fully coupled with the atmospheric, ocean, and wave models, use the following options in the **run_area** script in the ***jobs/[PROJECT NAME]*** directory when submitting a job.

   In **run_area**:

   ```
   ../../scripts/run_coamps -a1 -o1 -w1 -f1 [DTG]
   ```

   `-o3` can be used when running NCODA for the ocean assimilation (only after one update cycle).

5. To make sure the tropical cyclone vortex was initialized correctly in the atmospheric analysis, please view the **log.aanalysis** file in the ***runDir***. Do a search for "Meshes selected to follow TCs". If the analysis failed to find the tropical cyclone from the .**ngt[DTG]** file, the log file will mention a "BAD" tropical cyclone setup. The log file **tc_parm_[DTG]** contains the initial location and strength of the tropical cyclone being modeled, and the log files **tctrack1.[DTG]** and **tcwindr1.[DTG]** contain the location of the tropical cyclone vortex center and wind radii data for each hour in the COAMPS-TC simulation.

4.0 TECHNICAL REFERENCES

4.1 COAMPS Software Documentation

Allard, R., W.E. Rogers, S.N. Carroll, and K.V. Rushing, (2004). Validation Test Report for the Simulating Waves Nearshore Model (SWAN): Cycle III, Version 40.11. NRL Formal Report, NRL/FR/7322—04-10,070. Oceanography Division, Naval Research Laboratory, Stennis Space Center, Mississippi. 43 pp.

Allard, R., T.J. Campbell, T.A. Smith, T.G. Jensen, J.A. Cummings, S. Chen, J. Doyle, X. Hong, R.J. Small, S.N. Carroll, (2010). Validation Test Report for the Coupled Ocean/Atmosphere Mesoscale Prediction System, (COAMPS) Version 5.0. NRL Memo Report, NRL/MR/7320—10-9283, Oceanography Division, Naval Research Laboratory, Stennis Space Center, Mississippi. 160 pp.

Barron, C.N., A.B. Kara, P.J. Martin, R.C. Rhodes, and L.F. Smedstad, (2006). Formulation, implementation and examination of vertical coordinate choices in the Global Navy Coastal Ocean Model (NCOM). *Ocean Modelling*, **11:** 347-375.

Chen, S., J. Cummings, J. Doyle, R.H. Hodur, T. Holt, C. Liou, M. Liu, A. Mirin, J. Ridout, J.M. Schmidt, G. Sugiyama, and W.T. Thompson, (2003). "COAMPS Version 3 Model Description--General Theory and Equations". NRL/PU/7500--03-448, 145 pp.

Cummings, J.A., 2005: Operational multivariate ocean data assimilation. *Quart. J. Royal Met. Soc.*, Part C, **131**(613): 3583-3604.

Cummings, J. A. and S.N. Carroll, (2006). Software User Manual for the Navy Coupled Ocean Data Assimilation (NCODA) System. NRL Technical Report, NRL/MRY-001-06, Ocean Modeling Division, Naval Research Laboratory, Monterey, CA.

Martin, P.J., C.N. Barron, L.F. Smedstad, A.J. Wallcraft, R.C. Rhodes, T.J. Campbell, C. Rowley and S.N. Carroll, (2008a). Software Design Description for the Navy Coastal Ocean Model Version 4.0." NRL/MR/7320--08-9149, Ocean Modeling Division, Naval Research Laboratory, Stennis Space Center, MS.

Martin, P.J., C.N. Barron, L.F. Smedstad, T.J. Campbell, A.J. Wallcraft, R.C. Rhodes, C. Rowley, T.L. Townsend, and S.N. Carroll, (2008b). "User's Manual for the Navy Coastal Ocean Model (NCOM) Version 4.0." NRL/MR/7320--08-9151, Ocean Modeling Division, Naval Research Laboratory, Stennis Space Center, MS.

SWAN Team, (2010). SWAN Technical Document, SWAN Cycle III, version 40.51. Delft University, Environmental Fluid Dynamics section, Delft, The Netherlands.

Tolman, H. L., (2009). User manual and system documentation of WAVEWATCH III version 3.14. NOAA / NWS / NCEP / MMAB Technical Note **276**, 194 pp.+ Appendices.

4.2 General Technical References

Babanin, A. V., D. Chalikov, I. R. Young, and I. Savelyev, (2010). Numerical and laboratory investigation of breaking of steep two dimensional waves in deep water, *J. Fluid Mech.*, **644**: 433–463.

Blumberg, A.F. and G.L. Mellor, (1983). Diagnostic and prognostic numerical circulation studies of the South Atlantic Bight. *J. Geophys. Res.*, **88**: 4579-4592.

Blumberg, A.F. and G.L. Mellor, (1987). "A description of a three-dimensional coastal ocean circulation model." In: Three-Dimensional Coastal Ocean Models. N. Heaps, ed., American Union, New York, N.Y., p. 208.

Collins-Sussman, B., B.W. Fitzpatrick, and C.M. Pilato. "Version Control with Subversion." [Online]. Copyright © 2002, 2003, 2004, 2005, 2006, 2007 O'Reilly Media Inc, Sebastopol, CA. <http://subversion.tigris.org/>.

Geiszler, D., R. Wade, and J. Cook, (2003). The Naval Research Laboratory's Satellite Server (NRLSAT): An automated cloud verification system for COAMPS-OS® Battlespace Atmospheric and Cloud Impacts on Military Operations [BACIMO] Conference, Monterey, CA.

Hodur, R.M., (1997). The Naval Research Laboratory's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS). *Mon. Wea. Rev.,* **125**: 1414-1430.

Hogan, T., and T. Rosmond, (1991). The description of the Navy Operational Global Atmospheric Predictions System's spectral forecast model. *Mon.Wea. Rev.*, **119**: 1786-1815.

Janssen, P.A.E.M, (1991). Quasi-linear theory of wind-wave generation applied to wave forecasting. *J. Phys. Oceanogr.*, **21:** 1631–1642.

Kantha L.H. and C.A. Clayson, (2004). On the effect of surface gravity waves on mixing in the oceanic mixed layer. *Ocean Model* **6**:101–124.

Komen, G. J., S. Hasselmann, and K. Hasselmann, (1984) On the existence of a fully developed wind-sea spectrum, *J. Phys. Oceanogr*., **14:** 1271–1285.

Large, W.G., J.C. McWilliams, and S.C. Doney, (1994). Oceanic vertical mixing: a review and a model with a nonlocal boundary layer parameterization. *Rev. Geophys.*, **32**: 363-403.

Louis, J.F., M. Tiedtke, and J.F. Geleyn, (1981). A short history of the operational PBL-parameterization at ECMWF, Workshop on Planetary Boundary Layer Parameterization, 25-27 Nov. 1981, 59-79.

Martin, P.J., G. Peggion, and K.J. Yip, (1998). "A comparison of several coastal ocean models." NRL Report NRL/FR/7322--97-9692. Naval Research Laboratory, Stennis Space Center, MS., pp. 96.

Moon, I.-J., I. Ginis, T. Hara, (2004). Effect of surface waves on air–sea momentum exchange: II. Behavior of drag coefficient under tropical cyclones. *J. Atoms. Sci.*, **61**: 2334- 2348.

Pullen, J., J. Doyle, T. Haack, C. Dorman, R. Signell, C. Lee, (2007). Bora Event Variability and the Role of Air-Sea Feedback, *J. Geophys. Res*., **112** (C3): C03S18.

Rogers, W.E., A.V. Babanin, and D.W. Wang, (2011). Observation-consistent input and whitecapping-dissipation in a model for wind-generated surface waves: Description and simple calculations, *J. Atm. and Ocean Tech.*, (in press).

SWAN team, (2010). SWAN Scientific and Technical Documentation, SWAN Cycle III version 40.81, Delft University of Technology, http://www.swan.tudelft.nl , 118 pp.

Tolman, H. L., (1989). The numerical model WAVEWATCH: a third generation model for the hindcasting of wind waves on tides in shelf seas. *Communications on Hydraulic and Geotechnical Engineering*, Delft Univ. of Techn., ISSN 0169-6548, Rep. no. **89**(2): 72 pp.

Tolman, H. L., (1991a). A third-generation model for wind waves on slowly varying, unsteady and inhomogeneous depths and currents. *J. Phys. Oceanogr.*, **21**: 782-797.

Tolman, H. L., (1992). Effects of numerics on the physics in a third-generation wind-wave model. *J. Phys. Oceanogr*., **22**: 1095-1111.

5.0 NOTES

5.1 Acronyms and Abbreviations

Acronym	Description
3DVar	Three dimensional variable resolution
ADP	Atmospheric Data
AMSRE	Advanced Microwave Scanning Radiometer - Earth Observing System
ATSR	Along Track Scanning Radiometer
BC	Boundary Conditions
COAMPS	Coupled Ocean Atmosphere Mesoscale Prediction System
COAMPS-OS	Coupled Ocean Atmosphere Mesoscale Prediction System-On Scene
COAMPS-TC	Coupled Ocean Atmosphere Mesoscale Prediction System-Tropical Cyclone
CTD	Conductivity, Temperature and Depth
DBDBV	Digital Bathymetric Data Base- Variable resolution
DMSP	Defense Meteorological Satellite Program
DoD	Department of Defense
DSRC	DoD Supercomputing Resource Centers
DTG	Date-Time-Group
ERS-2	European Remote Sensing satellite 2
ESMF	Earth System Modeling Framework
GAC	Global Area Coverage
GB	Gigabyte
GFO	Geosat Follow-On
GNCOM	Global NCOM
GOES	Geostationary Operational Environmental Satellite
GUI	Graphical User Interface
GVC	General Vertical Coordinate
HPC	High Performance Computing
HPCMP	High Performance Computing Modernization Program
I/O	Input/Output
IC	Initial (Boundary) Conditions
IP	Internet Protocol
LAC	Local Area Coverage
LBC	Lateral Boundary Conditions
METOP	Polar Orbiting Meteorological satellite
MODAS	Modular Ocean Data Assimilation System
MSG	Meteosat Second Generation
MVOI	Multi-Variate Optimal Interpolation

Acronym	Description
MYL2/2.5	Mellor Yamada Level 2/2.5 vertical mixing scheme
NASA	National Aeronautical and Space Administration
NAVDAS	NRL Atmospheric Variational Data Assimilation System
NAVOCEANO	Naval Oceanographic Office
NCAR	National Center for Atmospheric Research
NCEP	National Centers for Environmental Prediction
NCODA	Navy Coupled Ocean Data Assimilation system
NCOM	Navy Coastal Ocean Model
NFS	Network File System
NOAA	National Oceanic and Atmospheric Administration
NOGAPS	Navy Operational Global Atmospheric Prediction System
NRL-MRY	Naval Research Laboratory, Monterey
NRL-SSC	Naval Research Laboratory, Stennis Space Center
OCNQC	Quality Controlled Ocean Data
PBS	Portable Batch System
POM	Princeton Ocean Model
RELO NCOM	Relocatable Navy Coastal Ocean Model
SGE	Sun Grid Engine
SPMD	Single Process, Multiple Data
SSH	Sea Surface Height
SST	Sea Surface Temperature
SWAN	Simulating WAves Nearshore
SWH	Significant Wave Height
SZM	Sigma/Z- level Model
TEDS	Tactical Environmental Database Services
TRMM	Tropical Rainfall Measuring Mission
WAM	WAve Model
WW3	WAVEWATCH III
XBT	Expendable BathyThermograph

Appendix A: /Jobs/ Directory Scripts

A-1 COAMPS Platform Setup Script (/jobs/setup_nrlssc)

```
#-------------------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/jobs/socal/setup_nrlssc $
# @(#)$Id: setup_nrlssc 429 2011-09-08 13:09:02Z campbell $
#-------------------------------------------------------------------------------#
#
# Script: setup_nrlssc
#       COAMPS site setup for NRLSSC platforms
#             pbs (Intel Xeon nodes with PBS batch system)
#             sge (Intel Xeon nodes with SGE batch system)
#             mac (Apple MacBook Pro, no batch)
#       To select a specific queue set the queue variable in the batch
#       job settings section.
#
# Purpose:
#   This script, sourced by run_coamps, adds site/platform specific settings
#   to the batch command file and sets platform specific variables that are
#   used in run_coamps.
#
# Require calling parameters:
#   NONE
#
# Global variables required:
#   ksh_executable     : path to Korn Shell 93 executable
#   USER               : user name (environment)
#   platform           : name of platform (host machine)
#   wave_type          : wave model type (swan, ww3 or none)
#   jobDir             : path to job directory (where run_coamps is invoked)
#   jobName            : name of batch job
#   cmdFile            : batch command file (with path)
#   cmdLog             : batch command log file (with path)
#   area               : name of simulation area/experiment
#   ddtg               : date-time-group of run
#   batch_nprocs       : number of processors for batch request
#   interactive_option : flag indicating job is interactive
#
# Global variables created:
#   runDir  : area ddtg run directory
#   prjDir  : projects area/experiment input file directory
#   srcDir  : full NFS path to the COAMPS source directory
#   batch   : batch submission command
#   + variables for path & archive commands
#
#-------------------------------------------------------------------------------#
```

```
name=setup_nrlssc


#-------------------------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------------------------#
check_variables $name \
ksh_executable USER platform jobDir jobName \
cmdFile cmdLog area ddtg batch_nprocs interactive_option \
|| exit 1
if [[ ! -e $cmdFile ]]
then
  print "$name: batch command file does not exist: $cmdFile" 2>&1; exit 1
fi
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------------------------#
if [[ $# != 0 ]]
then
  print "$name: incorrect number of input parameters" 2>&1; exit 1
fi
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# check for supported platform
#-------------------------------------------------------------------------------#
case $platform in
  pbs|sge|mac) ;;
  *) print "$name: unsupported platform: $platform" 2>&1; exit 1 ;;
esac
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# local variables
#-------------------------------------------------------------------------------#
# walltime (HH:MM:SS)
typeset wallTime="06:00:00"

# name of queue
typeset queue="standard"

# path to local COAMPS output data directory
typeset outDataDir="/u/COAMPS/scratch3/$USER"

# path to local COAMPS save data directory
typeset savDataDir="/u/COAMPS/data/$USER"
```

```
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# global variables
#-------------------------------------------------------------------------------#
# runDir: area ddtg run directory
# default value is "$outDataDir/$area/run/$ddtg"
runDir="$outDataDir/$area/run/$ddtg"

# prjDir: projects area input file directory
# default value is "$jobDir/../../projects/$area"
prjDir="$jobDir/../../projects/$area"

# srcDir: full NFS path to the COAMPS source directory
# override default with COAMPS_SRCDIR environment variable
case $platform in
  *) srcDir=${COAMPS_SRCDIR:="/u/COAMPS/src/dev/coamps4_esmf"} ;;
esac

# batch submission command
batch=qsub

# num_physical_cpus_per_node: number of physical cpus per node
# ncpus_per_node: number of mpi tasks per node
# ** must be <= number of physical cpus per node
case $platform in
  pbs) num_physical_cpus_per_node=12
       ncpus_per_node=12 ;;
  sge) num_physical_cpus_per_node=12
       ncpus_per_node=12 ;;
  mac) num_physical_cpus_per_node=2
       ncpus_per_node=2 ;;
esac

# mpirun commands
case $platform in
  *) mpicmd="mpirun -np"
     mpicmd_serial="mpirun -np 1" ;;
esac

# uncompress command
uncompressCmd=gunzip
#-------------------------------------------------------------------------------#


#===============================================================================#
# add batch directives to batch command file
#===============================================================================#
nnodes=$(( batch_nprocs / ncpus_per_node ))
if (( batch_nprocs - nnodes*ncpus_per_node > 0 ))
```

```
then
  nnodes=$(( nnodes + 1 ))
fi
case $platform in

pbs)
cat >> $cmdFile << End_Batch_Directives
#-------------------------------------------------------------------------------#
# batch directives
#-------------------------------------------------------------------------------#
#PBS -S ${ksh_executable}
#PBS -N ${jobName}
#PBS -o ${cmdLog}
#PBS -j oe
#PBS -m abe
#PBS -M ${USER}@mail7320
#PBS -l walltime=${wallTime}
#PBS -l nodes=${nnodes}:ppn=${ncpus_per_node}:${queue}
#PBS -q ${queue}
#-------------------------------------------------------------------------------#
End_Batch_Directives
;;

sge)
cat >> $cmdFile << End_Batch_Directives
#-------------------------------------------------------------------------------#
# batch directives
#-------------------------------------------------------------------------------#
#$ -S ${ksh_executable}
#$ -N ${jobName}
#$ -o ${cmdLog}
#$ -j y
#$ -m beas
#$ -M ${USER}@mail7320
#$ -q ${queue}
#$ -pe orte ${batch_nprocs}
#$ -l arch=lx24-amd64
#-------------------------------------------------------------------------------#
End_Batch_Directives
;;

mac)
if [[ $interactive_option == 0 ]]
then
  print "$name: platform $platform only supports interactive runs" 2>&1
  exit 1
fi
;;

esac
printf "\n\n" 1>> $cmdFile
```

```
#==================================================================#


#==================================================================#
# add paths and archive commands to batch command file
#==================================================================#
cat >> $cmdFile << End_Paths_And_Archive_Commands
#------------------------------------------------------------------#
# paths and archive commands
#------------------------------------------------------------------#

# path to COAMPS database
databaseDir=/u/COAMPS/input/database

# path to TC warning database
tcDir=/u/COAMPS/input/TCWarnings

# path to local f and adp data
fandaDir=/u/COAMPS/input/fanda

# path to local ocnqc data
ocnqcPublicDir=/u/COAMPS/input/ocnqc
ocnqcRstrctDir=/u/prob/ncoda/data/navoqc/navoqc_rstrct

# path to local gncom data
gncomDir=/u/NCOM

# path to local gwave data
gwaveDir=/u/WW3

# paths to model output data
dataDir=$outDataDir/$area
atmosDir=/u/COAMPS/scratch2/tasmith/Adr06valid/atmos
oceanDir=$outDataDir/$area/ocean
waveDir=$outDataDir/$area/wave
obkgdDir=$outDataDir/$area/obkgd
wbkgdDir=$outDataDir/$area/wbkgd
cplrDir=$outDataDir/$area/cplr

# run archive copy command and archive path
saveDir=$savDataDir/$area
archiveCopyCmd=cp
archiveDir=$savDataDir/$area

# NAVDAS stuff
navdasDir=$srcDir/atmosa
navdas_data=\$databaseDir/navdas_data/database
tfileDir=\$fandaDir
scrDir=
scrDir2=
expname=$area
```

```
#-------------------------------------------------------------------------#
End_Paths_And_Archive_Commands
printf "\n\n" 1>> $cmdFile
#=========================================================================#


#=========================================================================#
# add environment settings to batch command file
# *** these settings must be consistent with the COAMPS executables
# *** escape "\" is required so that variable substitution will occur
#     in cmdFile instead of in this script
#=========================================================================#
case $platform in

pbs)
cat >> $cmdFile << End_Environment_Settings
#-------------------------------------------------------------------------#
# environment settings
#-------------------------------------------------------------------------#

MPIDIR=/common/openmpi/pgi

PATH=/usr/local/bin:/usr/bin:/bin
PATH=\$MPIDIR/bin:\$PATH
export PATH

unset MPIDIR

#-------------------------------------------------------------------------#
End_Environment_Settings
;;

sge)
cat >> $cmdFile << End_Environment_Settings
#-------------------------------------------------------------------------#
# environment settings
#-------------------------------------------------------------------------#

MPIDIR=/common/openmpi/pgi
BSETUP=/u/gridengine/default/common/settings.sh

PATH=/usr/local/bin:/usr/bin:/bin
PATH=\$MPIDIR/bin:\$PATH
export PATH

. \$BSETUP

unset MPIDIR
unset BSETUP
```

```
#-------------------------------------------------------------------------#
End_Environment_Settings
;;

esac
printf "\n\n" 1>> $cmdFile
#=========================================================================#


#=========================================================================#
# batch job must use NFS paths
#=========================================================================#
case $platform in

pbs|sge)
if [[ $interactive_option != 1 ]]
then
case $(print $jobDir | cut -f2 -d/) in
  u|net|home) ;;
  *) echo "$myname: SGE job must be submitted from NFS directory" 1>&2 ;
     exit 1 ;;
esac
case $(print $outDataDir | cut -f2 -d/) in
  u|net|home) ;;
  *) echo "$myname: SGE job must use NFS path for output data directory" 1>&2 ;
     exit 1 ;;
esac
case $(print $savDataDir | cut -f2 -d/) in
  u|net|home) ;;
  *) echo "$myname: SGE job must use NFS path for save data directory" 1>&2 ;
     exit 1 ;;
esac
fi
;;

esac
#=========================================================================#


unset name
#-------------------------------------------------------------------------#
# end of script
#-------------------------------------------------------------------------#
```

A-2 COAMPS Experiment-Specific Setup Script (/jobs/setup_area)

```
#------------------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/jobs/socal/setup_area $
# @(#)$Id: setup_area 408 2011-07-14 16:12:02Z campbell $
#------------------------------------------------------------------------------#
#
# Script: setup_area
#
# Purpose:
#   This script, sourced by run_coamps, sets the simulation area specific
#   variables that are used in run_coamps.
#
# Require calling parameters:
#   NONE
#
# Global variables required:
#   jobDir  : path to job directory (where run_coamps is invoked)
#
# Global variables created:
#   area            : name of simulation area/experiment
#   jobName         : name of batch job
#   cmdFile         : batch command file (with path)
#   cmdLog          : batch command log file (with path)
#   wave_type       : wave model type (swan or ww3)
#   atmos_analysis_type : atmos analysis type (mvoi or 3dvar)
#   ocean_analysis_type : ocean analysis type (mvoi or 3dvar)
#   site            : name of site (corresponds to setup_${site}).
#   platform        : name of platform (host machine) -- used in setup_${site}
#   update_cycle    : number of hours for hindcast/forecast update cycle
#   fcst_length     : number of hours for forecast
#   couple_A2B      : coupled forecast with modelA-to-modelB exchange turned on/off
#   coupled_execution_mode: coupled forecast model execution mode (sequential or concurrent)
#   atmos_nproc(x,y) : number of tiles in x and y-direction for atmos
#   ocean_nproc(x,y) : number of tiles in x and y-direction for ocean
#    wave_nprocs    : number of processors for wave
#   atmosa_nprocs   : number of processors for atmos analysis (NAVDAS)
#   atmosa_nthreads  : number of OpenMP threads for atmos analysis (NAVDAS)
#   oceana_nprocs   : number of processors for ocean analysis (NCODA)
#   oceana_nthreads  : number of OpenMP threads for ocean analysis (NCODA)
#   ocards          : name of ocards file in project directory
#   xcards          : name of xcards file in project directory
#
# A section is available near the end of this function for creating user
# specific global variables.
#
#------------------------------------------------------------------------------#
name=setup_area
```

```
#-------------------------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------------------------#
check_variables $name jobDir || exit 1
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------------------------#
if [[ $# != 0 ]]
then
  print "$name: incorrect number of input parameters" 2>&1; exit 1
fi
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# area : name of simulation area/experiment
#     ** default value is "$(basename $jobDir)"
# jobName: name of batch job
#     ** default value is "$area.$ddtg"
# cmdFile: batch command script
#     ** default value is "$jobDir/cmd.$ddtg"
# cmdLog: batch command log file (with path)
#     ** default value is "$jobDir/log.$ddtg"
#-------------------------------------------------------------------------------#
area="$(basename $jobDir)"
jobName="$area.$ddtg"
cmdFile="$jobDir/cmd.$ddtg"
cmdLog="$jobDir/log.$ddtg"
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# site: name of site
#     ** corresponds to setup_${site} function
# platform: name of platform (host machine)
#     ** must be a supported platform in setup_${site}
#-------------------------------------------------------------------------------#
site=nrlssc
platform=pbs
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# update_cycle: number of hours for hindcast/forecast update cycle
# fcst_length: number of hours for forecast
#-------------------------------------------------------------------------------#
update_cycle=12
fcst_length=12
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
```

```
# couple_a2o=t/f - coupled forecast with atmos-to-ocean exchange turned on/off
# couple_o2a=t/f - coupled forecast with ocean-to-atmos exchange turned on/off
# couple_a2w=t/f - coupled forecast with atmos-to-wave  exchange turned on/off
# couple_w2a=t/f - coupled forecast with  wave-to-atmos exchange turned on/off
# couple_o2w=t/f - coupled forecast with ocean-to-wave  exchange turned on/off
# couple_w2o=t/f - coupled forecast with  wave-to-ocean exchange turned on/off
#-------------------------------------------------------------------------------#
couple_a2o=f
couple_o2a=f
couple_a2w=f
couple_w2a=f
couple_o2w=t
couple_w2o=t
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# wave_type: wave model type (swan or ww3)
#     ** wave model is active only when coupling with wave model is turned on
# coupled_execution_mode: coupled forecast model execution mode
#     concurrent - coupled forecast with execute in concurrent mode
#         ** atmos & ocean on different sets of processors
#         ** total # procs = # atmos procs + # ocean procs
#     sequential - coupled forecast with execute in sequential mode
#         ** atmos & ocean on same set of processors
#         ** total # procs = # atmos procs = # ocean procs
# atmos_nproc(x,y): number of tiles in x and y-direction for atmos
#     ** must be consistant the selected coupled execution mode
# ocean_nproc(x,y): number of tiles in x and y-direction for ocean
#     ** must be consistant the selected coupled execution mode
#  wave_nprocs    : number of processors for wave
#     ** must be consistant the selected coupled execution mode
#     ** SWAN tiling is 1D applied along the largest grid dimension
#-------------------------------------------------------------------------------#
wave_type=swan
coupled_execution_mode=sequential
atmos_nprocx=8  ; atmos_nprocy=8  ;
ocean_nprocx=8  ; ocean_nprocy=8  ;
 wave_nprocs=64
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# atmos_analysis_type: atmos analysis type
#     mvoi  - MVOI atmos analysis
#     3dvar - NAVDAS
# atmosa_nprocs: number of processors for atmos analysis
#     ** must be >= 4
# atmosa_nthreads: number of OpenMP threads for atmos analysis
#     ** will be capped to num_physical_cpus_per_node (set in setup_<site>)
#-------------------------------------------------------------------------------#
atmos_analysis_type=3dvar
atmosa_nprocs=16
```

```
atmosa_nthreads=8
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# ocean_analysis_type: ocean analysis type
#     mvoi  - NCODA MVOI
#     3dvar - NCODA 3DVAR
# ocean_analysis_update_opt: ocean analysis update option
#     update - apply analysis variable increment to model restart
#     relax  - apply analysis variable increment over relaxation period
#              specified in setup_nl_oparm (rlax_ts and rlax_uv)
# oceana_nprocs: number of processors for ocean analysis
# oceana_nthreads: number of OpenMP threads for ocean analysis
#     ** will be capped to num_physical_cpus_per_node (set in setup_<site>)
#-------------------------------------------------------------------------------#
ocean_analysis_type=3dvar
ocean_analysis_update_opt=update
oceana_nprocs=16
oceana_nthreads=8
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# host_atmos_type: type of host (global) atmos input files
#     nogaps36 - NOGAPS 36 character files
#     nogaps64 - NOGAPS 64 character files
#-------------------------------------------------------------------------------#
host_atmos_type=nogaps36
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# ocards: name of OCARDS file in project directory
#   ** will be copied into run directory and renamed OCARDS
# xcards: name of XCARDS file in project directory
#   ** will be copied into run directory and renamed XCARDS
#-------------------------------------------------------------------------------#
ocards=OCARDS.Gulf
xcards=XCARDS.default
#-------------------------------------------------------------------------------#

#-------------------------------------------------------------------------------#
# User specific global variables.
# *** be sure not to use typeset ***
#-------------------------------------------------------------------------------#
#example_global_var=xxyyzz
#-------------------------------------------------------------------------------#

unset name
#-------------------------------------------------------------------------------#
# end of script
#-------------------------------------------------------------------------------#
```

A-3 COAMPS Setup Script for Running on the DSRC (/jobs/setup_navy_dsrc)

```
#-------------------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/jobs/socal/setup_navy_dsrc $
# @(#)$Id: setup_navy_dsrc 438 2011-09-23 19:38:16Z campbell $
#-------------------------------------------------------------------------------#
#
# Script: setup_navy_dsrc
#       COAMPS site setup for the Navy DSRC platforms:
#            babbage (IBM P5+ with PBS)
#            davinci (IBM P6 with PBS)
#            einstein (Cray XT5 with PBS)
#
# Purpose:
#   This script, sourced by run_coamps, adds site/platform specific settings
#   to the batch command file and sets platform specific variables that are
#   used in run_coamps.
#
# Require calling parameters:
#   NONE
#
# Global variables required:
#   ksh_executable     : path to Korn Shell 93 executable
#   USER               : user name (environment)
#   platform           : name of platform (host machine)
#   wave_type          : wave model type (swan, ww3 or none)
#   jobDir             : path to job directory (where run_coamps is invoked)
#   jobName            : name of batch job
#   cmdFile            : batch command file (with path)
#   cmdLog             : batch command log file (with path)
#   area               : name of simulation area/experiment
#   ddtg               : date-time-group of run
#   batch_nprocs       : number of processors for batch request
#   interactive_option : flag indicating job is interactive
#
# Global variables created:
#   runDir  : area ddtg run directory
#   prjDir  : projects area/experiment input file directory
#   srcDir  : full NFS path to the COAMPS source directory
#   batch   : batch submission command
#   + variables for path & archive commands
#
#-------------------------------------------------------------------------------#
name=setup_navy_dsrc


#-------------------------------------------------------------------------------#
# check for required global variables
```

```
#-------------------------------------------------------------------------------#
check_variables $name \
ksh_executable USER platform jobDir jobName \
cmdFile cmdLog area ddtg batch_nprocs interactive_option \
|| exit 1
if [[ ! -e $cmdFile ]]
then
  print "$name: batch command file does not exist: $cmdFile" 2>&1; exit 1
fi
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------------------------#
if [[ $# != 0 ]]
then
  print "$name: incorrect number of input parameters" 2>&1; exit 1
fi
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# check for supported platform
#-------------------------------------------------------------------------------#
case $platform in
  babbage|davinci|einstein) ;;
  *) print "$name: unsupported platform: $platform" 2>&1; exit 1 ;;
esac
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# platform/user dependent paths (local variables)
#   parentDir  : path to parent COAMPS directory
#   scratchDir : path to user scratch directory
#-------------------------------------------------------------------------------#
case $platform in
  babbage ) typeset parentDir=/site/BEI/COAMPS
            typeset scratchDir=/scr/$USER ;;
  davinci ) typeset parentDir=/site/BEI/COAMPS
            typeset scratchDir=/scr/$USER ;;
  einstein) typeset parentDir=/scr/SITE/BEI/COAMPS
            typeset scratchDir=/scr/$USER ;;
esac
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# local variables
#-------------------------------------------------------------------------------#
```

```
# name of project account
typeset projAcct="HPCMO98520IN3"

# walltime (HH:MM:SS)
typeset wallTime="12:00:00"

# name of queue
typeset queue="standard"

# path to local COAMPS input data directory (location of fanda & ocnqc)
typeset inpDataDir="$scratchDir/COAMPS/data"

# path to local COAMPS output data directory
typeset outDataDir="$scratchDir/COAMPS/data"

# path to local COAMPS save data directory
typeset savDataDir="$scratchDir/COAMPS/save"
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# global variables
#-------------------------------------------------------------------------------#
# runDir: area ddtg run directory
# default value is "$outDataDir/$area/run/$ddtg"
runDir="$outDataDir/$area/run/$ddtg"

# prjDir: projects area input file directory
# default value is "$jobDir/../../projects/$area"
prjDir="$jobDir/../../projects/$area"

# srcDir: full NFS path to the COAMPS source directory
# override default with COAMPS_SRCDIR environment variable
srcDir=${COAMPS_SRCDIR:="$parentDir/coamps5_trunk"}

# batch submission command
batch=qsub

# num_physical_cpus_per_node: number of physical cpus per node
# ncpus_per_node: number of mpi tasks per node
# ** must be <= number of physical cpus per node
case $platform in
  babbage)  num_physical_cpus_per_node=16
            ncpus_per_node=16 ;;
  davinci)  num_physical_cpus_per_node=32
            ncpus_per_node=32 ;;
  einstein) num_physical_cpus_per_node=8
            ncpus_per_node=8  ;;
esac

# mpirun commands
```

```
case $platform in
  babbage)  mpicmd="poe -procs"
            mpicmd_serial="poe -procs 1" ;;
  davinci)  mpicmd="poe -procs"
            mpicmd_serial="poe -procs 1" ;;
  einstein) mpicmd="aprun -N $ncpus_per_node -n"
            mpicmd_serial="aprun -n 1" ;;
esac

# uncompress command
uncompressCmd=gunzip
#-------------------------------------------------------------------------------#


#===============================================================================#
# add batch directives to batch command file
#===============================================================================#
nnodes=$(( batch_nprocs / ncpus_per_node ))
if (( batch_nprocs - nnodes*ncpus_per_node > 0 ))
then
  nnodes=$(( nnodes + 1 ))
fi
case $platform in

babbage|davinci)
cat >> $cmdFile << End_Batch_Directives
#-------------------------------------------------------------------------------#
# batch directives
#-------------------------------------------------------------------------------#
#PBS -A ${projAcct}
#PBS -N $(print ${jobName} | cut -c1-15)
#PBS -o ${cmdLog}
#PBS -j oe
#PBS -m abe
#PBS -l walltime=${wallTime}
#PBS -l select=${nnodes}:ncpus=${ncpus_per_node}:mpiprocs=${ncpus_per_node}
#PBS -l place=scatter:excl
#PBS -q ${queue}
#-------------------------------------------------------------------------------#
ulimit -c unlimited 2>/dev/null #coredumpsize
ulimit -s unlimited 2>/dev/null #stacksize
ulimit -t unlimited 2>/dev/null #cputime
ulimit -d unlimited 2>/dev/null #datasize
ulimit -m unlimited 2>/dev/null #memoryuse
ulimit -a

End_Batch_Directives
;;

einstein)
cat >> $cmdFile << End_Batch_Directives
```

```
#-------------------------------------------------------------------------------#
# batch directives
#-------------------------------------------------------------------------------#
#PBS -A ${projAcct}
#PBS -N $(print ${jobName} | cut -c1-15)
#PBS -o ${cmdLog}
#PBS -j oe
#PBS -m abe
#PBS -l walltime=${wallTime}
#PBS -l mppwidth=${batch_nprocs}
#PBS -l mppnppn=${ncpus_per_node}
#PBS -q ${queue}
#-------------------------------------------------------------------------------#
ulimit -c unlimited 2>/dev/null #coredumpsize
ulimit -s unlimited 2>/dev/null #stacksize
ulimit -t unlimited 2>/dev/null #cputime
ulimit -d unlimited 2>/dev/null #datasize
ulimit -m unlimited 2>/dev/null #memoryuse
ulimit -a
export HOSTNAME=$platform

End_Batch_Directives
;;

esac
printf "\n\n" 1>> $cmdFile
#===============================================================================#


#===============================================================================#
# add paths and archive commands to batch command file
#===============================================================================#
cat >> $cmdFile << End_Paths_And_Archive_Commands
#-------------------------------------------------------------------------------#
# paths and archive commands
#-------------------------------------------------------------------------------#

# path to COAMPS database
databaseDir=$parentDir/database

# path to TC warning database
tcDir=$parentDir/TCWarnings

# path to local f and adp data
fandaDir=$inpDataDir/fanda

# path to local ocnqc data
ocnqcPublicDir=$inpDataDir/ocnqc
ocnqcRstrctDir=$inpDataDir/ocnqc

# path to local gncom data
```

```
gncomDir=$inpDataDir/gncom

# path to local gwave data
gwaveDir=$inpDataDir/gwave

# paths to model output data
dataDir=$outDataDir/$area
atmosDir=$outDataDir/$area/atmos
oceanDir=$outDataDir/$area/ocean
waveDir=$outDataDir/$area/wave
obkgdDir=$outDataDir/$area/obkgd
wbkgdDir=$outDataDir/$area/wbkgd
cplrDir=$outDataDir/$area/cplr

# run archive copy command and archive path
saveDir=$savDataDir/$area
archiveCopyCmd=cp
archiveDir=$savDataDir/$area

# NAVDAS stuff
navdasDir=$srcDir/atmosa
navdas_data=\$databaseDir/navdas_data/database
tfileDir=\$fandaDir
scrDir=
scrDir2=
expname=$area

#-------------------------------------------------------------------------#
End_Paths_And_Archive_Commands
printf "\n\n" 1>> $cmdFile
#=========================================================================#


#=========================================================================#
# add environment settings to batch command file
# *** these settings must be consistent with the COAMPS executables
# *** escape "\" is required so that variable substitution will occur
#     in cmdFile instead of in this script
#=========================================================================#
case $platform in

babbage|davinci)
cat >> $cmdFile << End_Environment_Settings
#-------------------------------------------------------------------------#
# environment settings
#-------------------------------------------------------------------------#

export MP_INFOLEVEL=1
export MP_STDOUTMODE=unordered
export MP_STDINMODE=all
export MP_LABELIO=no
```

```
#-------------------------------------------------------------------------------#
End_Environment_Settings
;;

einstein)
cat >> $cmdFile << End_Environment_Settings
#-------------------------------------------------------------------------------#
# environment settings
#-------------------------------------------------------------------------------#

export MPICH_PTL_SEND_CREDITS=-1
export MPICH_UNEX_BUFFER_SIZE=180M
export NAVBIN
#-------------------------------------------------------------------------------#
End_Environment_Settings
;;

esac
printf "\n\n" 1>> $cmdFile
#===============================================================================#


unset name
#-------------------------------------------------------------------------------#
# end of script
#-------------------------------------------------------------------------------#
```

A-4 ERDC DSRC Platform Setup Script (/projects/setup_erdc_dsrc)

```
#------------------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/jobs/socal/setup_erdc_dsrc $
# @(#)$Id: setup_erdc_dsrc 407 2011-07-14 05:38:20Z campbell $
#------------------------------------------------------------------------------#
#
# Script: setup_erdc_dsrc
#       COAMPS site setup for the ERDC DSRC platforms:
#            diamond (SGI Altix ICE with PBS)
#
# Purpose:
#   This script, sourced by run_coamps, adds site/platform specific settings
#   to the batch command file and sets platform specific variables that are
#   used in run_coamps.
#
# Require calling parameters:
#   NONE
#
# Global variables required:
#   ksh_executable     : path to Korn Shell 93 executable
#   USER               : user name (environment)
#   platform           : name of platform (host machine)
#   wave_type          : wave model type (swan, ww3 or none)
#   jobDir             : path to job directory (where run_coamps is invoked)
#   jobName            : name of batch job
#   cmdFile            : batch command file (with path)
#   cmdLog             : batch command log file (with path)
#   area               : name of simulation area/experiment
#   ddtg               : date-time-group of run
#   batch_nprocs       : number of processors for batch request
#   interactive_option : flag indicating job is interactive
#
# Global variables created:
#   runDir  : area ddtg run directory
#   prjDir  : projects area/experiment input file directory
#   srcDir  : full NFS path to the COAMPS source directory
#   batch   : batch submission command
#   + variables for path & archive commands
#
#------------------------------------------------------------------------------#
name=setup_erdc_dsrc


#------------------------------------------------------------------------------#
# check for required global variables
#------------------------------------------------------------------------------#
check_variables $name \
```

```
ksh_executable USER platform jobDir jobName \
cmdFile cmdLog area ddtg batch_nprocs interactive_option \
|| exit 1
if [[ ! -e $cmdFile ]]
then
  print "$name: batch command file does not exist: $cmdFile" 2>&1; exit 1
fi
#------------------------------------------------------------------------------#


#------------------------------------------------------------------------------#
# process input parameters
#------------------------------------------------------------------------------#
if [[ $# != 0 ]]
then
  print "$name: incorrect number of input parameters" 2>&1; exit 1
fi
#------------------------------------------------------------------------------#


#------------------------------------------------------------------------------#
# check for supported platform
#------------------------------------------------------------------------------#
case $platform in
  diamond) ;;
  *) print "$name: unsupported platform: $platform" 2>&1; exit 1 ;;
esac
#------------------------------------------------------------------------------#


#------------------------------------------------------------------------------#
# platform/user dependent paths (local variables)
#   parentDir  : path to parent COAMPS directory
#   scratchDir : path to user scratch directory
#------------------------------------------------------------------------------#
case $platform in
  diamond) typeset parentDir=/usr/local/usp/COAMPS
           typeset scratchDir=/work/$USER ;;
esac
#------------------------------------------------------------------------------#


#------------------------------------------------------------------------------#
# local variables
#------------------------------------------------------------------------------#
# name of project account
typeset projAcct="HPCMO98520IN3"

# walltime (HH:MM:SS)
typeset wallTime="12:00:00"
```

```
# name of queue
typeset queue="standard"

# path to local COAMPS input data directory (location of fanda & ocnqc)
typeset inpDataDir="$scratchDir/COAMPS/data"

# path to local COAMPS output data directory
typeset outDataDir="$scratchDir/COAMPS/data"

# path to local COAMPS save data directory
typeset savDataDir="$scratchDir/COAMPS/save"
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# global variables
#-------------------------------------------------------------------------------#
# runDir: area ddtg run directory
# default value is "$outDataDir/$area/run/$ddtg"
runDir="$outDataDir/$area/run/$ddtg"

# prjDir: projects area input file directory
# default value is "$jobDir/../../projects/$area"
prjDir="$jobDir/../../projects/$area"

# srcDir: full NFS path to the COAMPS source directory
# override default with COAMPS_SRCDIR environment variable
srcDir=${COAMPS_SRCDIR:="$parentDir/coamps5_trunk"}

# batch submission command
batch=qsub

# num_physical_cpus_per_node: number of physical cpus per node
# ncpus_per_node: number of mpi tasks per node
# ** must be <= number of physical cpus per node
case $platform in
  diamond) num_physical_cpus_per_node=8
           ncpus_per_node=8 ;;
esac

# mpirun commands
case $platform in
  diamond) mpicmd="mpiexec_mpt -np"
           mpicmd_serial="mpiexec_mpt -np 1" ;;
esac

# uncompress command
uncompressCmd=gunzip
#-------------------------------------------------------------------------------#
```

```
#==========================================================================#
# add batch directives to batch command file
#==========================================================================#
nnodes=$(( batch_nprocs / ncpus_per_node ))
if (( batch_nprocs - nnodes*ncpus_per_node > 0 ))
then
  nnodes=$(( nnodes + 1 ))
fi
case $platform in

diamond)
cat >> $cmdFile << End_Batch_Directives
#--------------------------------------------------------------------------#
# batch directives
#--------------------------------------------------------------------------#
#PBS -A ${projAcct}
#PBS -N $(print ${jobName} | cut -c1-15)
#PBS -o ${cmdLog}
#PBS -j oe
#PBS -m abe
#PBS -l walltime=${wallTime}
#PBS -l select=${nnodes}:ncpus=${ncpus_per_node}:mpiprocs=${ncpus_per_node}
#PBS -l place=scatter:excl
#PBS -q ${queue}
#--------------------------------------------------------------------------#
End_Batch_Directives
;;

esac
printf "\n\n" 1>> $cmdFile
#==========================================================================#


#==========================================================================#
# add paths and archive commands to batch command file
#==========================================================================
========#
cat >> $cmdFile << End_Paths_And_Archive_Commands
#--------------------------------------------------------------------------#
# paths and archive commands
#--------------------------------------------------------------------------#

# path to COAMPS database
databaseDir=$parentDir/database

# path to TC warning database
tcDir=$parentDir/TCWarnings

# path to local f and adp data
```

```
fandaDir=$inpDataDir/fanda

# path to local ocnqc data
ocnqcPublicDir=$inpDataDir/ocnqc
ocnqcRstrctDir=$inpDataDir/ocnqc

# path to local gncom data
gncomDir=$inpDataDir/gncom

# path to local gwave data
gwaveDir=$inpDataDir/gwave

# paths to model output data
dataDir=$outDataDir/$area
atmosDir=$outDataDir/$area/atmos
oceanDir=$outDataDir/$area/ocean
waveDir=$outDataDir/$area/wave
obkgdDir=$outDataDir/$area/obkgd
wbkgdDir=$outDataDir/$area/wbkgd
cplrDir=$outDataDir/$area/cplr

# run archive copy command and archive path
saveDir=$savDataDir/$area
archiveCopyCmd=cp
archiveDir=$savDataDir/$area

# NAVDAS stuff
navdasDir=$srcDir/atmosa
navdas_data=\$databaseDir/navdas_data/database
tfileDir=\$fandaDir
scrDir=
scrDir2=
expname=$area

#-------------------------------------------------------------------------#
End_Paths_And_Archive_Commands
printf "\n\n" 1>> $cmdFile
#=========================================================================#


#=========================================================================#
# add environment settings to batch command file
# *** these settings must be consistent with the COAMPS executables
# *** escape "\" is required so that variable substitution will occur
#     in cmdFile instead of in this script
#=========================================================================#
case $platform in

diamond)
cat >> $cmdFile << End_Environment_Settings
#-------------------------------------------------------------------------#
```

```
# environment settings
#-------------------------------------------------------------------------------#

export MPI_DSM_DISTRIBUTE=yes
export MPI_GROUP_MAX=64
export MPI_TYPE_MAX=65536
export MPICH_PTL_SEND_CREDITS=-1
export MPICH_UNEX_BUFFER_SIZE=180M
export NAVBIN

#-------------------------------------------------------------------------------#
End_Environment_Settings
;;

esac
printf "\n\n" 1>> $cmdFile
#===============================================================================#


unset name
#-------------------------------------------------------------------------------#
# end of script
#-------------------------------------------------------------------------------#
```

Appendix B: /Projects/ Directory Scripts

B-1 COAMNL Namelist Setup Script (/projects/setup_nl_coamnl)

```
#------------------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_coamnl $
# @(#)$Id: setup_nl_coamnl 408 2011-07-14 16:12:02Z campbell $
#------------------------------------------------------------------------------#
#
# Function: setup_nl_coamnl
#
# Purpose:
#   Create the coamnl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#------------------------------------------------------------------------------#
function setup_nl_coamnl {


#------------------------------------------------------------------------------#
# check for required global variables
#------------------------------------------------------------------------------#
check_variables $0 \
update_cycle fcst_length gridnl_atmos atmos_analysis_type \
host_atmos_type host_atmos_prl \
|| return 1
#------------------------------------------------------------------------------#


#------------------------------------------------------------------------------#
# process input parameters
#------------------------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#------------------------------------------------------------------------------#


#------------------------------------------------------------------------------#
# delta:   model coarse domain (large) time step (second)
#------------------------------------------------------------------------------#
typeset delta=120.0
#------------------------------------------------------------------------------#
```

```
#-------------------------------------------------------------------------------#
# idbms:   Specify format of input/output files
#   1 : read old NOGAPS and write old COAMPS files (36 characters, sequential)
#   2 : read new NOGAPS and write new COAMPS files (64 characters, direct access)
#   3 : read new NOGAPS and write old COAMPS files (36 characters, sequential)
#   4 : read old NOGAPS, new COAMPS (iupd=2), and write new COAMPS files
#                                   (64 characters, direct access)
#   5 : read old NOGAPS, old COAMPS (iupd=2), and write new COAMPS files
#                                   (64 characters, direct access)
#-------------------------------------------------------------------------------#
case $host_atmos_type in
  nogaps36) typeset idbms=4 ;;
  nogaps64) typeset idbms=2 ;;
esac
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# sigma levels (selection based on kka in atmos gridnl)
#-------------------------------------------------------------------------------#
typeset dsigma_array
case ${gridnl_atmos.kka} in
30) dsigma_array=( \
      7500.0 5800.0 4200.0 2500.0 1000.0 \
      1000.0  750.0  750.0  750.0  750.0 \
       750.0  750.0 1000.0 1000.0 1000.0 \
      1000.0  800.0  800.0  800.0  600.0 \
       400.0  300.0  200.0  140.0   90.0 \
        60.0   40.0   30.0   20.0   20.0 \
   ) ;;
40) dsigma_array=( \
      7500.0 5800.0 4200.0 2500.0 1000.0 \
      1000.0  750.0  750.0  750.0  750.0 \
       750.0  750.0 1000.0 1000.0 1000.0 \
      1000.0  800.0  800.0  800.0  600.0 \
       400.0  300.0  200.0  140.0  120.0 \
       110.0  100.0  100.0  100.0  100.0 \
       100.0  100.0   90.0   80.0   70.0 \
        60.0   40.0   30.0   20.0   20.0 \
   ) ;;
60) dsigma_array=( \
      2500.0 2500.0 2500.0 1000.0 1000.0 \
       500.0  500.0  500.0  500.0  500.0 \
       500.0  500.0  500.0  500.0  500.0 \
       500.0  500.0  500.0  500.0  500.0 \
       500.0  500.0  500.0  500.0  500.0 \
       500.0  500.0  500.0  500.0  500.0 \
       500.0  500.0  500.0  500.0  500.0 \
       500.0  500.0  500.0  500.0  500.0 \
       400.0  300.0  200.0  140.0  120.0 \
```

```
       110.0  100.0  100.0  100.0  100.0 \
       100.0  100.0   90.0   80.0   70.0 \
        60.0   40.0   30.0   20.0   20.0 \
   ) ;;
*) #***** unsupported *****
  error_msg "$0: no sigma levels setup for kka=${gridnl_atmos.kka}"
  return 1 ;;
esac
typeset dsigma=$(print ${dsigma_array[*]})
dsigma=${dsigma//+([[:space:]])/,}
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# atmos analysis dependent settings
# *** must be consistent with lm in atmosnl setup ***
#-------------------------------------------------------------------------------#
typeset l3dvar loi lpseud pr
case $atmos_analysis_type in
  3dvar) # NAVDAS: host pressure levels
    l3dvar=t; loi=f; lpseud=f;
    pr=$(print ${host_atmos_prl[*]})
  ;;
  mvoi)  # MVOI: 16 levels
    l3dvar=f; loi=t; lpseud=t;
    pr="10 20 30 50 70 100 150 200 250 300 400 500 700 850 925 1000"
  ;;
esac
pr=${pr//+([[:space:]])/,}
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# atmospheric boundary condition pressure levels
#-------------------------------------------------------------------------------#
typeset prbc=$(print ${host_atmos_prl[*]})
prbc=${prbc//+([[:space:]])/,}
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# namelist head
#-------------------------------------------------------------------------------#
cat << end_namelist
&coamnl
end_namelist
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# model configuration options
```

```
#
# idbms:   Specify format of input/output files
# lmpi2:   MPI-1/MPI-2 parameter (t = use MPI-2; f = use MPI-1)
# lcoda    t: use CODA sst and sea ice analysis
#          f: use NOGAPS
# kgetbc:  frequency for reading coarse domain boundary tendencies
# lanalbc: t: use NOGAPS analysis fields for coarse domain boundary tendencies
#          f: use NOGAPS forecast fields for coarse domain boundary tendencies
# delta:   model coarse domain (large) time step (second)
# ktaust:  starting time of forecast (hr, min,sec)
# ktauf:   ending time of forecast (hr, min,sec)
# ncast_start : nowcast start parameter
# ncast_end   : nowcast end parameter
# ncast_cycle : nowcast cycle parameter
# dsigma:  1D array (dimension of kka); vertical grid spacing (meter)
# l2way    t: feedback the nest domain info back to its parent domain
#-------------------------------------------------------------------------------#
cat << end_namelist

 idbms         = ${idbms},
 lmpi2         = f,
 lcoda         = t,
 kgetbc        = 6,
 lanalbc       = t,

 delta         = ${delta},
 ktaust        = 0, 0, 0,
 ktauf         = ${fcst_length}, 0, 0,
                 ${fcst_length}, 0, 0,
                 ${fcst_length}, 0, 0,
                 ${fcst_length}, 0, 0,

 ncast_start    = 0,
 ncast_end      = 0,
 ncast_cycle    = 0,

 dsigma        = ${dsigma},

 l2way         = t,

end_namelist
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# data assimilation options
#
# iupd:    data assimilation identifier
#          0 - none
#          1 - full update
#          2 - incremental update (it is recommended to use 2 for data assimilation)
```

```
# itauin:  frequency for writing coarse mesh boundary tendencies
# itaus:   start time for analysis boundary condition
# itauf:   end time for writing coarse mesh boundary tendencies
# icycle:  data assimilation cycle (hr)
# l3dvar:  3dvar flag
#-------------------------------------------------------------------------------#
cat << end_namelist

 iupd          = 2, 2, 2, 2, 2, 2, 2,
 itauin        = 6, 0, 0,
 itaus         = 0, 0, 0,
 itauf         = ${fcst_length}, 0, 0,
 icycle        = ${update_cycle},

 l3dvar        = ${l3dvar},

end_namelist
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# objective analysis options
#
# loi:     T - multivariate optimum interpolation analysis
# loimf:   T - inner mesh multivariate optimum interpolation analysis
# lqanl:   T - pressure level specific humidity analysis using Cressman scheme
# ltanl:   T - pressure level temperature analysis using Cressman scheme
# lpseud:  T - output pseudo observation base on NOGAPS fields
# ivol,jvol: overlapping volume size in x & y directions for MVOI
# ibog1,jbog1: size of pseudo-obs analysis volumes in x &y directions for MVOI;
#              used if lpseud = t;
# prbc: pressure levels (hPa) for analysis boundary conditions
# pr: pressure levels (hPa)
#-------------------------------------------------------------------------------#
cat << end_namelist

 loi           = ${loi},
 loimf          = t,
 lqanl          = t,
 ltanl          = t,
 lpseud         = ${lpseud},
 ivol           = 10,31,151,151,
 jvol           = 10,31,151,133,
 ibog1          = 22,
 jbog1          = 22,

 prbc          = ${prbc},
 pr            = ${pr},

end_namelist
#-------------------------------------------------------------------------------#
```

```
#-------------------------------------------------------------------------------#
# input and output options
#
# ldiagt:  tketotal and thetotal mpi diagnostic print
# lprint:  T - more diagnostic prints
# kprnta:  print frequency for forecast fields (hr, min, sec)
# ifsave:  model output options
#          1 - specify save time
#          2 - specify save frequency (hr, min, sec)
# ksaves:  time (hr,min,sec) to save sigma data output
# isavefrq: output frequency (hr,min,sec) to save sigma data
# ksavea:  output frequency for writing model restart file (hr, min,sec)
#          it is also used to restart a model run at a nonzero time
# ksavmsp: output frequency to compute and write out
#          the maximum lowest sigma wind speed. Defaults is -1,0,0.
# ksavpcp: frequency (h,m,s) to tip the precipitation bucket
# npfil:   name of file containing 2D horizontal level plotting info
# xsfil:   name of file containing 2D cross-sectional plotting info
#-------------------------------------------------------------------------------#
cat << end_namelist

 ldiagt        = t,
 lprint        = f,
 kprnta         = 999, 0, 0,
 ifsave         = 2,
 ksaves         = ${update_cycle}, 0, 0,
 isavefrq       = 1, 0, 0,
 ksavea         = ${update_cycle}, 0, 0,
 ksavmsp        = -1, 0, 0,
 ksavpcp        = 999, 0, 0,
               999, 0, 0,
               999, 0, 0,
               999, 0, 0,
 npfil          = 'OCARDS',
 xsfil          = 'XCARDS',

end_namelist
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# numeric options
#
# iadvct:  1: 4th order; 2: 2nd order advection
#          3: 2nd order with 2nd oder upstream for theta
#          4: spectrl
#          5: 2nd order flux
#          6: mixed; 2nd order advective and flux
# lspong:  T - damp (u,v,w,theta) to smoothed values in the top nrdamp levels
```

```
#         F - do not do above
# rdtime:  time scale to damping u,v,w,theta in sponge layer
# nrdamp:  number of upper model levels that are in the sponge layer
# lralee:  raleigh damping layer
# robert:  coefficient for coupling the three time levels; used to filter high
#         frequency oscillations
# lddamp:  T - divergence damping on
# ldiff:   T - numerical diffusion on
# iupbc:   iupbc=0 : no radiation upper boundary condition (default)
#         iupbc=1 : radiation upper boundary condition on for idealized cases
#         iupbc=2 : radiation upper boundary condition on for real data cases
#-----------------------------------------------------------------------------#
cat << end_namelist

 iadvct        = 2,
 lspong        = t,
 rdtime        = 240.0,
 nrdamp        = 4,
 lralee        = f,
 robert        = 0.2,
 lddamp        = f,
 ldiff         = t,
 iupbc         = 0,

end_namelist
#-----------------------------------------------------------------------------#


#-----------------------------------------------------------------------------#
# physics options
#
# lcupar:  T - Kain-Fritsch convective parameterization on
# lcuppr:  T - Kain-Fritsch diagnostics print on
# icup :   3 - new KF
# lmoist:  T - allow for moisture
# lice:    T - ice physics on
# lrad:    T - radiation on
# lflux:   T - surface fluxes on
# lsfcen:  T - allow surface fluxes and radiation to affect the ground t and q
#         idealized:  T- initialize pressure, or potential temperature, or
#                 add perturbation to horizontal wind
# dxmeso:  upper limit (in meter) of horizontal resolution below which the vertical
#         advection of tke, raindrops and snow are included in prognostic euq.
# nradtyp: radiation type (1=; 2=Fuliou;)
#-----------------------------------------------------------------------------#
cat << end_namelist

 lcupar        = t,
 lcuppr        = f,
 icup          = 3, 3, 3, 3, 3, 3, 3,
 lmoist        = t,
```

```
 lice          = t,
 lrad          = t,
 lflux         = t,
 lsfcen        = t,
 dxmeso        = 10000.0,
 nradtyp       = 2,

end_namelist
#-----------------------------------------------------------------------------#


#-----------------------------------------------------------------------------#
# turbulence options
#
# ltke:    T - subgrid scale mixing on
# iashsgm: 1 - use deformation field for subgrid mixing
#          2 - use tke prediction
# iashsm:  1 - Ri dependent (Mellor & Yamada, 74)
#          2 - sm=sm0, sh=sh0
#          3 - sm=sm0, sh=(1.0+2.0 al2/al1)*sm0
#          4 - sm=sm0, sh=2.13*sm0
#          5 - Ri dependent (Mellor & Yamada, 82)
# iamxgl:  mixing length identifier (1,2,3,4,5), see user's guide table D-1
# sh0:     constant; used to compute the tke coefficient sh
# sm0:     constant; used to compute the tke coefficient sm
#-----------------------------------------------------------------------------#
cat << end_namelist

 ltke          = t,
 iahsgm        = 0,
 iashsm        = 5,
 iamxgl        = 5,
 sh0           = 0.675,
 sm0           = 0.5,

end_namelist
#-----------------------------------------------------------------------------#


#-----------------------------------------------------------------------------#
# tropical cyclone & moving nest options
# ** specify moving nests in gridnl.atmos via lnmove parameter
#    (e.g., lnmove=f,t,t : nest 1 stationary, nests 2 & 3 move)
#
# ltctrk: T - tropical cyclone tracking turned on
# ntcmov: id of nest that will track tropical cyclone
# tcid: tropical cyclone identifier
#       fourth column in ngt files in adp directory is the tropical cyclone
#       number, fifth column is location: L = Atlantic, E = Eastern Pacific,
#       W = Western Pacific.  E.g., Frances in the Atlantic '06L' (6th storm).
# movemx: maximum number of grid points per iteration that nest can move
```

```
#-------------------------------------------------------------------------------#
cat << end_namelist

 ltctrk        = f,
 ntcmov        = 3,
 tcid          = '06L',
 movemx        = 1, 9, 12,

end_namelist
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# namelist tail
#-------------------------------------------------------------------------------#
cat << end_namelist
/
end_namelist
#-------------------------------------------------------------------------------#


}
#-------------------------------------------------------------------------------#
# end of function
#-------------------------------------------------------------------------------#
```

B-2 NCOM Parameter Namelist Setup Script (/projects/setup_nl_oparm)

```
#-------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_oparm $
# @(#)$Id: setup_nl_oparm 452 2011-11-03 17:12:53Z campbell $
#-------------------------------------------------------------------#
#
# Function: setup_nl_oparm
#
# Purpose:
#   Create the oparm (NCOM parmlst) namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   1 (in) : nest id
#
#-------------------------------------------------------------------#
function setup_nl_oparm {


#-------------------------------------------------------------------#
# check for require global variables
#-------------------------------------------------------------------#
check_variables $0 \
area ddtg update_cycle fcst_length gridnl_ocean \
|| return 1
#-------------------------------------------------------------------#


#-------------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------------#
if [[ $# != 1 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
typeset nest=$1 #nest id
if (( $nest > ${gridnl_ocean.nnest} ))
then
  error_msg "$0: input nest=$nest > ocean gridnl nnest=${gridnl_ocean.nnest}"
  return 1
fi
#-------------------------------------------------------------------#


#-------------------------------------------------------------------#
# user setup
#-------------------------------------------------------------------#
# set timestep (s) for coarse nest
typeset -E dti_base=30.0
```

```
# set feedback to coarse nests (0 = none; 1= T & S)
typeset feedbk=0

# open bc flag (=0 no open bndys; =1 use IC; =2 use input file; =3 parent nest)
typeset indobc=2
if [[ $nest > 1 ]]
then
  indobc=3
fi

# specified tidal forcing at open boundaries of parent nest (0=no, 1=yes)
typeset indtide=1
if [[ $nest > 1 ]]
then
  indtide=0
fi

# cyclic boundaries for parent nest (0=none, 4=x, 5=y, 6=both x & y)
typeset indcyc=0
if [[ $nest > 1 ]]
then
  indcyc=0
fi
#-----------------------------------------------------------------------#


#-----------------------------------------------------------------------#
# process grid info from ocean gridnl
#-----------------------------------------------------------------------#
# fix nest ratio to 3
typeset -i nst3=1
if [[ $nest > 1 ]]
then
  nst3=3
fi

# fix nest timestep ratio to 3
typeset -i nst4=1
if [[ $nest > 1 ]]
then
  nst4=3
fi

# compute nest timestep
typeset -i n
typeset -E dti=$dti_base
for ((n=2; n<=$nest; n++))
do
  dti=$(( dti / nst4 ))
done
```

```
# some nesting parameters (nsto)
typeset nst2=${gridnl_ocean.npgrid[$nest]}
typeset nst5=${gridnl_ocean.ii[$nest]}
typeset nst6=${gridnl_ocean.jj[$nest]}

# adjust for ncom nest indexing
if [[ $nst2 > 1 ]]
then
  nst5=$(( nst5 + (nst3+1)/2 ))
  nst6=$(( nst6 + (nst3+1)/2 ))
fi
#-----------------------------------------------------------------------#


#-----------------------------------------------------------------------#
# data-time info
#-----------------------------------------------------------------------#
typeset idate=$(print $ddtg | cut -c1-8)
typeset itime=$(print $ddtg | cut -c9-10)000000
#-----------------------------------------------------------------------#


#-----------------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------------#
cat << end_namelist
&parmlst

!  names for model, experiment, and domain.
!  note that these are not currently used for anything significant.
     modelo = 'NCOM'               ,! name of model.
     expto  = '${area}'            ,! name of experiment.
     domain = '${area}.nest${nest}'  ,! name of domain.

!  run control (iruno,lruno):
     idate=${idate} ,!1  date for start of run (YYYYMMDD).
     itime=${itime} ,!2  time for start of run (HHMMSSCC).
     rstart=$(get_ocean_restart_flag any) ,!1  restart from (standard) restart file.
     batch=T     ,!2  run in batch mode (no interaction).
     tothrs=${fcst_length},!1  total length of run (hrs).

!  output control (outo):
     out = 12.0   ,!1  frequency for output of restart file (h).
           1.0    ,!2  frequency for output of 3D fields (h).
           1.0    ,!3  frequency for output of surface fields (h).
           ${update_cycle),!4frequency for output of NFS restart file (h).
           0.0    ,!5  frequency for output of kinetic energy (h).
           0.0    ,!6  frequency for output of values at single pts (h).
           0.0    ,!7  frequency for output of transports (h).
           0.0    ,!8  frequency for output of water-mass volumes (h).
```

```
      0.0    ,!9  frequency for output of budgets (h).
      0.0    ,!10 frequency for output of wave forcing (h).
      0.0    ,!11 not currently used.
      0.0    ,!12 not currently used.

!  output control for surface fields output file (iouto).
     inde2  = 1    ,!1  include surface elevation:    =0 no, =1 yes.
     indvb2 = 1    ,!2  include barotropic transport: =0 no, =1 yes.
                    !   note:  = (depth-ave  u and v velocity)*(depth).
     indv2  = 1    ,!3  include surface velocity:     =0 no, =1 yes.
     indt2  = 1    ,!4  include surface temperature:  =0 no, =1 yes.
     inds2  = 1    ,!5  include surface salinity:     =0 no, =1 yes.
     inda2  = 1    ,!6  include surface windstress:   =0 no, =1 yes.

!  output control for 3-D fields output file (iouto).
     inde3  = 1    ,!11 include surface elevation:    =0 no, =1 yes.
     indvb3 = 1    ,!12 include barotropic transport: =0 no, =1 yes.
                    !   note:  = (depth-ave  u and v velocity)*(depth).
     indv3  = 1    ,!13 include 3-D u and v velocity: =0 no, =1 yes.
     indw3  = 1    ,!14 include 3-D vertical velocity:=0 no, =1 yes.
     indt3  = 1    ,!15 include 3-D temperature:      =0 no, =1 yes.
     inds3  = 1    ,!16 include 3-D salinity:         =0 no, =1 yes.
     inda3  = 1    ,!17 include surface atm forcing:  =0 no, =1 yes.
                    !   note:  this includes surface windstress, solar,
                    !   net surface heat flux (longwave + latent +
                    !   sensible), and evaporation - precipitation.

!  run and output control for nowcast-forecast simulations (infso).
     idatnow=${idate}  ,!2  nowcast date, date for end of hindcast,
                       !    and start of forecast (YYYYMMDD).
     itimnow=${itime}  ,!3  nowcast time, time for end of hindcast,
                       !    and start of forecast (HHMMSSCC).
     irs_out  = 1    ,!11 output NFS restart file only once ( =1)
                     !   or at regular intervals ( =2).
                     !   1 = output NFS restart file only once at
                     !       the elapsed time set in outo(4,nesto).
                     !   2 = output NFS restart file at regular
                     !       intervals defined by outo(4,nesto).
     irs_date = 2    ,!12 flag to indicate type of date tag to use
                     !   for NFS restart file:
                     !   0 = none (standard restart filename).
                     !   1 = date_time = YYYYMMDD_HHMMSSCC
                     !   2 = dtg = YYYYMMDDHH
     irs_mean = 0    ,!13 save values in array "rmean" to restart file.
                     !   rmean contains mean/climate T,S fields.
                     !   this is needed if these fields vary in time.
                     !   0 = do not save rmean in restart file.
                     !   1 = save rmean in restart file.
     irs_fmt  = 1    ,!14 format of restart *.B file being read.
                     ! 0 = unformatted w elapsed time (old-style).
                     ! 1 = formatted w date-time of fields (new-style).
```

```
                ! this is to allow reading old-style
                ! unformatted restart files.  only new-style
                ! formatted *.B files are written.
    irs_rset = 0    ,!15 reset elapsed time & iteration count to
                ! zero when reading old-style unformatted
                ! *.B restart file.  if this is done, the
                ! initial date-time MUST be set to the time
                ! of the restart fields.  note that new-style
                ! formatted *.B restart files are not affected
                ! by this since they store the date-time of the
                ! restart fields and will set the elapsed time
                ! based on the difference between the date-time
                ! of the restart fileds and the initial
                ! date-time specified for the run.
                ! 0 = do not reset elapsed time.
                ! 1 = reset when reading restart file.
    ioutdate = 1    ,!21 flag to indicate type of date-time tag to use
                ! for surface and 3D output (outsf,out3d) files:
                ! 0 = none (std output files).
                ! 1 = date_time = YYYYMMDD_HHMMSSCC
                ! 2 = dtg = YYYYMMDDHH
                ! 3 = date_hr  = YYYYMMDD_HH   for hindcast,
                !     date_tau = YYYYMMDD_tHHH for forecast.
                ! 4 = dtg     = YYYYMMDDHH      for hindcast,
                !     dtg_tau = YYYYMMDDHH_tHHH for forecast.
                ! 5 = dtg_time = YYYYMMDDHH_MM for hindcast,
                !     dtg_tau  = YYYYMMDDHH_tHHHMM forecast.
                ! note:  time format may be expanded if
                ! output frequency exceeds that for specified
                ! format.
    ioutnow = 0     ,!22 flag to write surface and 3D output files
                ! only before ( =-1) or after ( =1) nowcast
                ! time.
                ! 0 = output at all times.
                ! -1 = output only before and at nowcast time.
                ! 1 = output only at and after nowcast time.
    irlx2now = 1    ,!31 flag to turn off relaxation of SST and SSS
                ! after the nowcast time (if set =0).
                ! this flag will turn SST-SSS relax off if it
                ! is on, but will not turn it on if it is off.
    irlx3now = 1    ,!32 flag to turn off relaxation of 3D T and S
                ! after the nowcast time (if set =0).
                ! this flag will turn 3-D T-S relax off if it
                ! is on, but will not turn it on if it is off.

! physical options (phyo):
    mode   =3,!1  flag to indicate type of model run:
                !  =3 full calculation, all fields updated.
                !  =4 diagnostic calculation (T & S held fixed).
    indcor =2,!2  flag to Coriolis value to be used:
                !  =0 zero.
```

```
            !=1 spatially constant value based on mean latitude.
            ! =2 spatially variable computed from local latitude.
indden =3,!3  density formula:  =1 Frederich-Levitus; =3 Mellor-UNESCO
            !   note:  use Mellor ( =3) for deep sigma layers or
            !   if using partial cells (with NCOM version 3+).
indadv =1,!4  momentum advection: =0 off; =1 on.
indadvr=2,!5  scalar advection: =0 off; =1 on; =2 on and use FCT.
            !   note:  FCT avoids overshoots but is costly for memory
            !   and running time.
indxk  =2,!6  horizontal diffusion scheme:
            !   =0 none;  =1 grid-cell Re number;  =2 Smagorinsky.
indzk  =5,!7  vertical mixing scheme: =1 constant; =2 MYL2;
            ! =4 MYL2.5 ~ as described by Mellor and Yamada (1982);
            ! =5 MYL2.5 ~ as described by Kantha and Clayson (2004).
            ! if indzk=4,5, CPP macro MYL2P5 must be defined for
            ! the compile.
indtkes=2,!8  treatment of surface BC for TKE for MYL2.5 mixing:
            ! =1 use value at surface computed from windstress.
            ! =2 use flux at surface computed from windstress.
            !note:  flux BC ( =2) provides TKE from wave breaking,
            ! which can increase mixing near surface, but does not
            ! affect ML depth much.
indlxts=0,!9  relax T and S 3-D fields each timestep:
            ! =0 no relaxation.
            ! =1 relax to T & S fields input as mean or climate
            ! fields in file otscl (these are fixed in time).
            ! =2 relax to time-varying T and S fields input in file otsf.
            ! =3 add TS increments from NCOM_TSINC over the first
            ! rlax_ts hours of the integration
            ! =4 add TS increments from NCOM_TSINC plus the OTSF
            ! minus restart TS difference
indlxuv=0,!15 relax U and V 3-D fields each timestep:
            ! =0 no relaxation.
            ! =1 relax to U & V fields input as mean or climate
            ! fields in file ouvcl (these are fixed in time).
            ! =2 unsupported - does nothing
   ! =3 add UV increments from NCOM_UVINC over the first
            ! rlax_uv hours of the integration
   ! =4 add UV increments from NCOM_UVINC plus the OVELF
            ! minus restart UV difference
indext =1,!10 solar extinction profile:
            ! =0 no solar extinction.
   ! =1 2-band exponential approximation to Jerlov types.
   ! =4 derive extinction profiles from chl input data (Morel).
   ! =5 derive extinction from k490 input data (Morel).
indtype=2,!11 Jerlov optical type to use when indext=1:
               ! =1 I; =2 IA; =3 IB; =4 II; =5 III
               ! note:  no temporal or spatial variability.
indbio =0,!12 bio model:  =0 none;
               ! =1 use default (very simple) 4-component bio model.
               ! note:  the default model mainly exists to act as a
```

```
      ! "placeholder" for a more precise or elaborate model.
    indice =0,!13 simple parameterization of an ice model:  =0 none;
      !   =1 reduce surface windstress at ice-covered locations.
    bclinic=T,!1  baroclinic pressure gradients:
      !   T calculate baroclinic pressure gradient.
      !   F set baroclinic pressure gradients =0.
    curved =T,!2  horizontal advection grid curvature term:
      !   T calculate.
      !   F do not calculate.
      !   should be set =T for curvilinear grids.
    noslip =T,!3  noslip momentum lateral bc for horizotal diffusion:
       !   T noslip lateral BC.
       !   F free-slip later BC.
       !   note: only applies for 2nd-order adv with indxk=1,2.
    sigdif =T,!4  for horizontal mixing of scalars on the sigma grid,
       !   subtract off mean field.  also applied for high-order
       !   correction for 3rd- or 4th-order scalar advection.
    largmix=T,!5  use Large et al. background vertical mixing w MYL2.
                !   note:  this allows for mixing near base of ML
                !   up to critical Ri numbers of 0.7 and increases
                !   the depth of mixing.
    wetdry =F,!6  use wetting/drying (not yet implemented).
    tidpot =T,!7  turn on the local tidal potential.  tidal
       !   potential forcing is currently provided for
       !   consitutents K1 O1 P1 Q1 K2 M2 N2 S2 MF MM.
       !   constituents must be specified in an input file.
       !   note:  tidal potential forcing is needed if
       !   running with tides in a large-deep domain.
       !   it does not matter in shallow, coastal domains.

!  physical parameters (phyo):
    rho0=1025.0   ,!1  reference density for seawater (kg/m3).
    g=9.80        ,!2  gravitational constant (m/s2).
    cp=3994.0     ,!3  specific heat for seawater (joules/kg/degC).
    ramphrs=0.0,!4  ramphrs - length of initial linear ramp (hrs).
       !   note:  this is used to ramp up forcing slowly
       !   for cold starts.  set =0 for warm starts.
    xkmin=10.0,!5  minimum hor momentum diffusion coef in x (m2/s).
       !   note:  used for cell-Re (indxk=1) and Smagorinski
            !   (indxk=2) horizontal diffusion.
    ykmin=10.0    ,!6  minimum hor momentum diffusion coef in y (m2/s).
            !   note:  used for cell-Re (indxk=1) hor diffusion.
    xkre=100.0    ,!7  maximum horizontal grid-cell Renolds Number.
            !   note:  used for cell-Re (indxk=1) hor diffusion.
    smag=0.1,!8  scaling constant for Smagorinsky horizontal diffusion.
       !   note:  used for Smagorinsky (indxk=2) hor diffusion.
    prnxi=1.0,!9  horizontal inverse Prandtl Number.
       !   note:  used for horizontal diffusion (indxk=1,2).
    zkmmin=0.1e-4,!10 minimum vertical diffusion coef for momentum (m2/s).
    zkhmin=0.1e-4 ,!11 minimum vertical diffusion coef for scalars (m2/s).
            !   note:  zkmmin and zkhmin are used to specify
```

```
              !  background or ambient vertical mixing rates.
     zkre=20.0     ,!12 maximum vertical grid-cell Renolds Number.
              !  note:  used for MYL2 mixing only (indzk=2).
     cbmin=0.0025  ,!13 minimum bottom drag coefficient.
     botruf1=0.010 ,!14 bottom roughness (m) default value (constant).
              !  note:  spatially variable values can be used.
     rlax_ts=6.0   ,!15 timescale for relaxation of deep T and S (hrs).
              !  note:  if this is set =0, a spatially variable
              !  3-D field of relaxation weights is read in from
              !  an input file (must be provided), which allows
              !  different specification of the relaxation
              !  timescale at every model grid point, e.g.,
              !  stronger relaxation near open boundaries or
              !  weaker relaxation near the coast.
              !  when indlxts=3 or 4 this specifies the insertion
              !  period for T/S increments
     rlax_uv=6.0   ,!18 timescale for relaxation of deep U and V (hrs).
     rlax_ds=5.0   ,!16 depth scale for relaxation of deep T and S (m).
              !  this is used in conjuction with rlax_ts to
              !  specify the temporal relaxation profile.
     b1_myl2=15.00 ,!17 Mellor Yamada Level 2 dissipation constant.
              !  note:  increasing this constant decreases the
              !  dissipation of TKE and increases mixing and MLD.

!  numerical options (numo):
     itermom=1     ,!1  number of iterations of momentum equations.
     indbaro=2,!2  solution method for free-surface mode:
          !  =1 full explicit (requires very small timestep).
               !  =2 semi-implicit.
     indsolv=1,!3  solver for semi-implicit free-surface mode:
               !  =1 cgssor (pre-condioned conjugate gradient).
               !  =2 sorcyc2 (SOR).
               !  =3 sorcyc2 (Jacobi - symmetric soln).
     indrag =2,!4  bottom drag calculation used for model simulations
               !  with just a single layer:  =1 explicit;
               !  =2 psuedo implicit;  =3 fully implicit.
     ifdadrh=3     ,!5  hor  adv of scalars:   =2 2nd order; =3 upw3.
     ifdadrv=3     ,!6  vert adv of scalars:   =2 2nd order; =3 upw3.
     ifdaduh=3     ,!7  hor  adv of momentum:  =2 2nd order; =3 upw3.
     ifdaduv=3     ,!8  vert adv of momentum:  =2 2nd order; =3 upw3.
               !  note:  upw3 refers to 3rd-order upwind scheme.
     ifdpgrd=4     ,!9  horizontal baroclinic presssure-gradient calc:
               !  =2 2nd order;  =4 4th order.
     ifdcor =4     ,!10 horizonal Coriolis interpolation:
               !  =2 2nd order;  =4 4th order.
     botrun =F     ,!1 truncate bottom grid cell to bathymetry.
               !  note:  this is not implemented here.  ncom
               !  version 3+ allows partial bottom cells, but
               !  this is handled during setup, not at run time.
     forward=T     ,!2 use forward scheme on 1st timestep.
     vector =T     ,!3 use vectorizable subroutines.
```

```
    shrnkwp=F      ,!4 use shrinkwrapping in x coordinate direction.
                   !  note:  set =T only when running with ipr=1.

!  numerical parameters (rnumo):
    dti    =${dti},!1 internal timestep (s).
    dte    =5.0,!2 external timestep for split-explicit (s) (not used).
    asf    =0.05,!3 Asselin filter coefficient.
    eg1    =0.5,!4 temporal wt for implicit surface elevation at n+1.
    eg2    =0.0,!5 temporal wt for implicit surface elevation at n  .
    vg1    =0.5,!6 temporal wt for momentum in continuity eqn at n+1.
    vg2    =0.0,!7 temporal wt for momentum in continuity eqn at n  .
    cb_filt=-0.125,!8 checkerboard mixing filter max value:
                   !  =  0.0   no checkerboard mixing filter is used.
                   !  =  0.125 use 5-pt hanning filter.
                   !  = -0.125 use 9-pt box filter.
                   !  note:  9-pt filter (used when cb_filt < 0)
                   !  provides max supression of checkerboard mixing.
    cb_dep =30.0,!9 depth of application of checkerboard filter (+m).
           !  note:  strength of checkerboard filter is reduced
           !  linearly from surface to depth defined by cd_dep.

!  surface forcing options and parameters (isbco, sbco):
    indsbc=1,!1 atm forcing:  =0 none (all turned off); =1 turned on.
    indatp=1,!2 surf atm press:  =0 none; =1 use osflx input data file;
         !  =2 use coupled atm model; =3 use COAMPS native grid files.
    indtau=1  ,!3 wind stress:  =0 none; =1 use osflx input data file;
       !  =2 use coupled atm model; =3 use COAMPS native grid files.
    indsft=5,!4 surf heat flux:  =0 none; =1 use osflx input data file;
        !  =2 use coupled atm model; =3 use COAMPS native grid files;
        !  =4 calc w bulk formula from COAMPS fields;
              !  =5 calc w bulk formula from data in osflx input file.
    indsfs=5,!5 surf salt flux:  =0 none; =1 use osflx input data file;
        !  =2 use coupled atm model; =3 use COAMPS native grid files;
        !  =4 calc w bulk formula from COAMPS fields;
        !  =5 calc w bulk formula from data in osflx input file.
    indsol=1,!6 solar flux:  =0 none; =1 use osflx input data file;
        !  =2 use coupled atm model; =3 use COAMPS native grid files.
    indcld=0  ,!7 flag for internal calculation of solar radiation from
              !  cloud cover input (not currently used).
    indsst=0  ,!8 sst relaxation to specified values:  =0 no; =1 yes.
    indsss=0,!9 sss relaxation to specified values:=0 no; =1 yes and sss
      !  being relaxed to is at same times as SST being relaxed to
      !  and is in same input file; =2 yes but SSS being relaxed
      !  to is in different file from SST being relaxed to and may
      !  be at different times.
    indsruf=2,!10 surface roughness:  =0 zero; =1 input data file (not used);
      !  =2 estimate from windstress using Charnock relation.
    rlaxsst=1.0,!1 rate of relaxation of sst to specified values (m/d).
    rlaxsss=1.0,!2 rate of relaxation of sss to specified values (m/d).
    charnok=2000.0,!3 Charnock constant for calc surf roughness from windstr.
```

```
!  lateral boundary options (iobco):
     indcyc =${indcyc},!1 use cyclic bndys:  =0 none; =4 x; =5 y; =6 both x & y.
     indtide=${indtide},!2 include specified tidal forcing at open boundaries:
             !  =0 no;  =1 yes.
     indobc =${indobc} ,!3 obc:  =0 no open bndys; =1 use IC; =2 use input file;
             !  =3 internal nest - obtain from grid in which nested.
     indobe =2 ,!4 obc for elevation (e):  =1 clamped; =2 Flather.
     indobvb=2 ,!5 obc for tangential depth-averaged transport:
             !  =1 zero gradent;  =2 Orlanski; =3 Ko adv.
     indobu =3 ,!6 obc for normal 3D velocity:  =1 intermal model calc;
             !  =2 Orlanski; =3 internal model calc plus normal adv;
             !  =4 Ko adv.
     indobv =2 ,!7 obc for tangential 3D velocity:  =1 zero gradient;
             !  =2 Orlanski; =3 Ko adv.
     indobr =2 ,!8 obc for scalars:  =1 put external specified value at bndy;
             !  =2 Orlanski; =3 adv, =4 Ko adv; =5 2D radiation.

!  lateral boundary parameters (obco):
     rlxobvb=8.0  ,!1 relaxation_timescale/dt for depth-averaged velocity.
     rlxobv =8.0  ,!2 relaxation_timescale/dt for 3D velocity.
     rlxobr =8.0  ,!3 relaxation_timescale/dt for scalar fields.

!  river inflows (rivo):
     indriv =1  ,!1 turn rivers on/off: =0 off; =1 on.
     indrivr=1  ,!2 number of scalar river inflow values specified.

!  grid nesting parameters (nsto):
     nst = ${nest},!1  grid number.
         ${nst2},!2  index of grid in which nested.
         ${nst3},!3  nesting ratio.
         ${nst4},!4  timestep ratio.
         ${nst5},!5  i-index of lower left corner of grid nested in.
         ${nst6},!6  j-index of lower left corner of grid nested in.
         ${feedbk},!7  feedback to coarse grid:  =0 none; =1 T and S;

!  diagnostics (diago):
     indiag=2   ,!1 =0 no diagnostics, > 0 higher level of diagnostics.
     indsym=0   ,!2 =0 no symmetry constraints and checks
              !  =1 symmetry constraints and 4-fold symmetry checks
              !  =2 symmetry constraints and 8-fold symmetry checks
     locate=F   ,!1 print out names of subroutines entered.

/
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-3 NCOM Setup Namelist Script (/projects/setup_nl_setupl)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_setupl $
# @(#)$Id: setup_nl_setupl 452 2011-11-03 17:12:53Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_setupl
#
# Purpose:
#   Create the setupl namelist for NCOM setup and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_setupl {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 databaseDir gridnl_ocean || return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters: should be 0"; return 1
fi
#-----------------------------------------------------------------#

#-----------------------------------------------------------------#
# Namelist description:
#-----------------------------------------------------------------#
#       Name          Type                Description
#   -------------   --------    ---------------------------------
#   lo              integer     total number of vertical layers + 1
#   lso             integer     number of sigma layers + 1
#   nro             integer     number of scalar fields.
#   nqo             integer     number of turbulence fields.
#   ntypo           integer     number of optical water types.
#   ntco            integer     number of tidal constituents in tidal bndy cond.
#   nobmaxo         integer     max number of open boundary pts.
```

```
#  nrivo          integer     number of (horizontal) river input pts.
#
#  dmin           real        minimum depth on the model grid (m, positive up)
#  dmax           real        maximum depth on the model grid (m, positive up)
#  dztop          real        thickness of model surface layer (m)
#  strfac         real        log stretching factor for the vertical grid
#  llog           integer     index where log stretching starts
#
#  dmaxout        real        maximum depth (m, positive up) for z-levels
#                               written to OZOUT file
#
#  lwav           integer     number of vertical depths for wave forcing input
#  dmax_wav       real        maximum depth of wave forcing input vertical grid (m, positive up)
#  dztop_wav      real        thickness (m) of top layer of wave forcing vertical grid
#                               wave forcing vertical grid is written to OZWAV file
#
#  dtf            real        time step for coamps surface forcing (s)
#  dtt            real        time step for coamps sst (s)
#  dtcyc          real        length of coamps analysis/forecast cycle (s)
#  dtmin          real        min forecast time for using coamps fields (s)
#  inesta         integer     nest grid for coamps surface forcing
#  sftopt         integer     surface heat flux option
#                               0 = off
#                               1 = calc w bulk formula from osflx (indsft=5)
#  sfsopt         integer     surface salt flux option
#                               0 = off
#                               1 = calc w bulk formula from osflx (indsfs=5)
#  idbms          integer     format of coamps file input
#                               1 = 36 characters, sequential access
#                               2 = 64 characters, direct access
#  ifcast         integer     option for coamps surface forcing
#                               0 = use coamps operational fields
#                               1 = use coamps reanalysis fields
#
#  startatrest    logical     set initial conditions to rest
#  writeosstf     logical     write an SST relaxation file
#  writeotsf      logical     write a 3D T/S relaxation file
#  writeotsu      logical     write a 3D T/S/U/V relaxation file
#  initialtide    logical     add tide heights/transports to initial fields
#
#  bathyfile      character   path to netcdf bathymetry file
#  riverfile      character   path to Paul Martin format river DB
#  tidefile       character   path to Paul Martin format OSU tide DB
#
#  rlx_bdy        real        boundary width scale in gridpoints
#                              if rlx_bdy <=1, relaxation is set
#                               through the interior
#                              if rlx_bdy > 1, relaxation weight
#                               decreases away from the boundary
#  rlx_tscli      real        time scale in days in interior and sfc
#  rlx_tsclb      real        ... and in the boundary at depth
```

```
#   rlx_zscl       real       vertical structure scale depth in m
#
#   dbcrit         real       if >0, critical slope limit
#   lmedian        logical    if true, run 9-pt filter on bathy
#   nofilt         integer    if >0, number of bathy smooth passes
#   lconsea        logical    if true, identify single contiguous water region
#
#-----------------------------------------------------------------#
# max number of nests (value set in oceanf/include/PARAM.h)
typeset nnestmx=7

# lo = kkom+1
typeset lo=$((${gridnl_ocean.kkom}+1))

# lso = kkosm
typeset lso=$((${gridnl_ocean.kkosm}+0))


#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
&setupl

 lo      = ${nnestmx}*${lo},
 lso     = ${nnestmx}*${lso},
 nro     = ${nnestmx}*2,
 nqo     = ${nnestmx}*2,
 ntypo   = ${nnestmx}*1,
 ntco    = ${nnestmx}*8,
 nobmaxo = ${nnestmx}*4000,
 nrivo   = ${nnestmx}*200,

 dmin    = -2.0,
 dmax    = -1261.6873,
 dztop   = 1.0,
 strfac  = 1.153018475,
 llog    = 1,

 dmaxout = -3000.0,

 lwav      = 50
 dmax_wav  = -300.0,
 dztop_wav = 0.1,

 dtf    = ${nnestmx}*3600,
 dtt    = 3600,
 dtcyc  = 43200,
 dtmin  = 3600,
 inesta = 3,
 sftopt = 1,
```

```
 sfsopt = 1,
 idbms  = 2,
 ifcast = 1,

 startatrest = .false.,
 initialtide = .true.,
 writeosstf  = .false.,
 writeotsf   = .false.,
 writeotsu   = .false.,

 bathyfile = 'DB2',
 riverfile = '${databaseDir}/river/rivers.dat',
 tidefile  = '${databaseDir}/tide/tide_egb.dat',

 rlx_bdy   =  10.0,
 rlx_tscli =  30.0,
 rlx_tsclb =   5.0,
 rlx_zscl  = 100.0,

 dbcrit    = 0.0,
 lmedian   = .false.,
 nofilt    = 1,
 lconsea   = .true.,

/
end_namelist
#-----------------------------------------------------------------#

}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-4 ESMF Configuration Setup Script (/projects/setup_esmf_config)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_esmf_config $
# @(#)$Id: setup_esmf_config 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_esmf_config
#
# Purpose:
#   Create the COAMPS ESMF config input and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_esmf_config {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 \
ddtg fcst_length coupled_execution_mode \
couple_a2o couple_o2a couple_o2w couple_w2o couple_a2w couple_w2a \
atmos_enabled atmos_nprocs atmosDir \
ocean_enabled ocean_nprocs oceanDir \
wave_enabled wave_nprocs waveDir wave_type \
obkgdDir wbkgdDir cplrDir fake_atmos_enabled \
|| return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# user setup
#-----------------------------------------------------------------#
#
```

```
# cpl_sec: coupling time-interval in (integer) seconds for coupled forecast
# (must be divisible by the time-step of each coupled forecast model)
#
typeset -i cpl_sec=360

#
# ocean_export_init_only=t - only initial ocean fields exchanged (couple_o2a=t)
#                    =f - time-varying ocean fields exchanged (couple_o2a=t)
#
typeset ocean_export_init_only=f

#
# rmpmsk_output_flag=t - output data files for regrid diagnostics
#                 =f - no diagnostic output
#
typeset rmpmsk_output_flag=t
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# check for consistency with atmos/ocean/wave setup
#-----------------------------------------------------------------#
typeset -E dta=10000
typeset -E dto=10000
typeset -E dtw=10000
typeset -E rta=0
typeset -E rto=0
typeset -E rtw=0

# get atmos timestep and ratio wrt coupling interval
if [[ $atmos_enabled == [tT] ]]
then
  dta=$(shvar delta $prjDir/setup_nl_coamnl) \
  || { error_msg "$0:$LINENO: get atmos timestep failed"; return 1; }
  rta=$(( fmod(cpl_sec,dta) ))
fi

# get ocean timestep and ratio wrt coupling interval
if [[ $ocean_enabled == [tT] ]]
then
  dto=$(shvar dti_base $prjDir/setup_nl_oparm) \
  || { error_msg "$0:$LINENO: get ocean timestep failed"; return 1; }
  rto=$(( fmod(cpl_sec,dto) ))
fi

# get wave timestep and ratio wrt coupling interval
if [[ $wave_enabled == [tT] ]]
then
  dtw=1
  case $wave_type in
    swan) dtw=$(shvar dtw $prjDir/setup_swan_input) \
```

```
        || { error_msg "$0:$LINENO: get wave timestep failed"; return 1; } ;;
    ww3 ) dtw=$(shvar dtw $prjDir/setup_ww3_grid) \
        || { error_msg "$0:$LINENO: get wave timestep failed"; return 1; } ;;
  esac
  rtw=$(( fmod(cpl_sec,dtw) ))
fi

# check that coupling interval is divisible by model timesteps
if [[ $rta != 0 || $rto != 0 || $rtw != 0 ]]
then
  error_msg "$0:$LINENO: coupling interval is not divisible by model timesteps" \
            "atmos timestep: $dta" \
            "ocean timestep: $dto" \
            "wave  timestep: $dtw" \
            "coupling interval: $cpl_sec"
  return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# coupled execution mode
#-----------------------------------------------------------------#
typeset concurrent_cpl_mode
case $coupled_execution_mode in
  concurrent) concurrent_cpl_mode=t ;;
  sequential) concurrent_cpl_mode=f ;;
esac
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# ocean background component
#-----------------------------------------------------------------#
typeset gocn_step=$(nlvar host_tauinc $prjDir/setup_nl_hostnl) \
|| { error_msg "$0:$LINENO: get global ocean step failed"; return 1; }
typeset obkgd_type=tendency
typeset obkgd_gridnl_file=gridnl.atmos
typeset -i obkgd_tendency_interval=$(( gocn_step * 3600 ))
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# wave background component
#-----------------------------------------------------------------#
typeset wbkgd_type=constant
typeset wbkgd_gridnl_file=gridnl.atmos
typeset -i wbkgd_tendency_interval=$(( gocn_step * 3600 ))
#-----------------------------------------------------------------#
```

```
#-----------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------#
cat << end_namelist
#####################################################################
#####################################################################
# COAMPS ESMF configuration file
#####################################################################
#####################################################################

#####################################################################
# Base date-time group
#####################################################################

base_dtg:   ${ddtg}

#####################################################################
# Start time
#####################################################################

start_hour: 0
start_min:  0
start_sec:  0

#####################################################################
# End time
#####################################################################

end_hour:   ${fcst_length}
end_min:    0
end_sec:    0

#####################################################################
# Coupling interval -- must be divisible by the time step
# of each enabled component
#####################################################################

cpl_hour:   0
cpl_min:    0
cpl_sec:    ${cpl_sec}

#####################################################################
# Execution mode
#####################################################################

concurrent_cpl_mode: ${concurrent_cpl_mode}

#####################################################################
# Atmos component
#####################################################################
```

```
atmos_num_pets: ${atmos_nprocs}
atmos_gridnl_file: gridnl.atmos
atmos_data_dir: ${atmosDir}
atmos_export_avg_enable: t
atmos_outff_flag: f
atmos_outff_interval: 3600
atmos_fake_enabled: ${fake_atmos_enabled}

######################################################################
# Ocean component
######################################################################

ocean_num_pets: ${ocean_nprocs}
ocean_gridnl_file: gridnl.ocean
ocean_data_dir: ${oceanDir}
ocean_outff_flag: f
ocean_outff_interval: 3600
ocean_nest1_to_egrid_flag: t

######################################################################
# Wave component
######################################################################

wave_num_pets: ${wave_nprocs}
wave_gridnl_file: gridnl.wave
wave_data_dir: ${waveDir}
wave_outff_flag: f
wave_outff_interval: 3600

######################################################################
# Coupling
#
# pmsl   : air_pressure_at_sea_level
# tauu   : surface_downward_eastward_stress
# tauv   : surface_downward_northward_stress
# hfns   : surface_downward_heat_flux
# mfns   : surface_downward_moisture_flux
# risw   : isotropic_shortwave_radiance_in_air
# sst    : sea_surface_temperature
# wndu   : eastward_10m_wind
# wndv   : northward_10m_wind
# chnk   : wave_induced_charnock_parameter
# wvst   : surface_total_wave_induced_stress
# wvsu   : surface_eastward_wave_induced_stress
# wvsv   : surface_northward_wave_induced_stress
# ssh    : sea_surface_height_above_sea_level
# sscu   : surface_eastward_sea_water_velocity
# sscv   : surface_northward_sea_water_velocity
# sdcu   : eastward_stokes_drift_current
# sdcv   : northward_stokes_drift_current
# wbcu   : eastward_wave_bottom_current
```

```
# wbcv   : northward_wave_bottom_current
# wbcf   : wave_bottom_current_radian_frequency
# wsuu   : eastward_wave_radiation_stress
# wsuv   : eastward_northward_wave_radiation_stress
# wsvv   : northward_wave_radiation_stress
# wsgu   : eastward_wave_radiation_stress_gradient
# wsgv   : northward_wave_radiation_stress_gradient
#
# wave_export_chnk_type: 1=w2a_janssen, 2=w2a_moon
#
#####################################################################

cpl_atmos_to_ocean: ${couple_a2o}
cpl_atmos_to_ocean_field_list: pmsl tauu tauv hfns mfns risw
cpl_ocean_to_atmos: ${couple_o2a}
cpl_ocean_to_atmos_field_list: sst
cpl_atmos_to_wave: ${couple_a2w}
cpl_atmos_to_wave_field_list: wndu wndv
cpl_wave_to_atmos: ${couple_w2a}
cpl_wave_to_atmos_field_list: chnk
cpl_ocean_to_wave: ${couple_o2w}
cpl_ocean_to_wave_field_list: ssh sscu sscv
cpl_wave_to_ocean: ${couple_w2o}
cpl_wave_to_ocean_field_list: sdcu sdcv wbcu wbcv wbcf wsgu wsgv

wave_export_chnk_type: 2

#####################################################################
# Ocean background -- these settings are used when the
# ocean-to-atmos or ocean-to-wave coupling is enabled.
#
# Wave background -- these settings are used when the
# wave-to-atmos or wave-to-ocean coupling is enabled.
#
# Available background types:
#    constant : Background values are constant in space
#               and time.
#    analysis : Background values are obtained from the
#               tau0 analysis and remain fixed in time.
#               When this option is selected the data
#               directory must be provided.
#    tendency : Background values are obtained from input
#               tendency files.  When this option is
#               selected the data directory and tendency
#               interval must be provided.
#####################################################################

obkgd_type: ${obkgd_type}
obkgd_gridnl_file: ${obkgd_gridnl_file}
obkgd_data_dir: ${obkgdDir}
obkgd_tendency_interval: ${obkgd_tendency_interval}
```

```
wbkgd_type: ${wbkgd_type}
wbkgd_gridnl_file: ${wbkgd_gridnl_file}
wbkgd_data_dir: ${wbkgdDir}
wbkgd_tendency_interval: ${wbkgd_tendency_interval}

#####################################################################
# Optional parameters for controlling the calculation of
# the regrid sparse matrices.
#   x2y_regrid_interp_method:
#     - type of interp method
#     - 1=bilinear (default), 2=bicubic
#   x2y_regrid_extrap_nnbrl:
#     - multiplier for number nnbr search levels for extrap
#     - >=1 (default=2)
# x2y = a2o | o2a | a2w | w2a | o2w | w2o
#####################################################################

a2o_regrid_interp_method: 1
a2o_regrid_extrap_nnbrl: 2
o2a_regrid_interp_method: 1
o2a_regrid_extrap_nnbrl: 2
a2w_regrid_interp_method: 1
a2w_regrid_extrap_nnbrl: 2
w2a_regrid_interp_method: 1
w2a_regrid_extrap_nnbrl: 2
o2w_regrid_interp_method: 1
o2w_regrid_extrap_nnbrl: 2
w2o_regrid_interp_method: 1
w2o_regrid_extrap_nnbrl: 2

#####################################################################
# Optional parameters for regrid IO and diagnostics
#   coupler_data_dir: path for coupler IO (default='.')
#   x2y_rmpmsk_output_flag:
#     - t : output data files for regrid diagnostics
#     - f : no diagnostic output (default)
# x2y = a2o | o2a | a2w | w2a | o2w | w2o
#####################################################################

coupler_data_dir: ${cplrDir}
a2o_rmpmsk_output_flag: ${rmpmsk_output_flag}
o2a_rmpmsk_output_flag: ${rmpmsk_output_flag}
a2w_rmpmsk_output_flag: ${rmpmsk_output_flag}
w2a_rmpmsk_output_flag: ${rmpmsk_output_flag}
o2w_rmpmsk_output_flag: ${rmpmsk_output_flag}
w2o_rmpmsk_output_flag: ${rmpmsk_output_flag}


#############################################################
#####################################################################
```

```
# end COAMPS ESMF configuration file
#########################################################################
#########################################################################
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-5 AERONL Namelist Creation Script (/projects/setup_nl_aeronl)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_aeronl $
# @(#)$Id: setup_nl_aeronl 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_aeronl
#
# Purpose:
#   Create the aeronl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_aeronl {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 databaseDir || return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# namelist description:
#-----------------------------------------------------------------#
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
&aeronl

 aerotme0 = 1, 0, 0,
```

```
 emstime  = 1, 0, 0,
          1, 0, 1,
 kccfrq   = 1, 0, 0,
 kcctyp   = 2,
 lwaycc   = 3,
 nadvtyp  = 1,
 ngrdems  = 1,
 ntems    = 2,

 cnthgt   =  10.00, 30.00, 55.00,
 cntlat   =  30.75, 30.75, 30.75,
 cntlon   =  46.45, 46.45, 46.45,
 emstrng  = 200.00,  0.00,
          200.00,  0.00,
          300.00,  0.00,

 ccycle   = f,
 dustflg  = t,           !run dust component
 lmsload  = t,           !have dust load output
 loptic   = t,           !have dust optical field output
 trceflg  = f,           !do not run passive tracer

 pthdust  = '${databaseDir}/dustsrc/'

/
end_namelist
#-----------------------------------------------------------------#

}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-6 Atmospheric Namelist Creation Script (/projects/setup_nl_atmosnl)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_atmosnl $
# @(#)$Id: setup_nl_atmosnl 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_atmosnl
#
# Purpose:
#   Create the atmosnl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_atmosnl {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 \
ddtg atmos_nprocx atmos_nprocy atmos_analysis_type \
host_atmos_dimx host_atmos_dimy host_atmos_prl \
|| return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# namelist description:
#-----------------------------------------------------------------#
# cdtg: date-time-group (character*10: yyyymmddhh)
#
# ndxnam: number of processors used in the x direction
# ndynam: number of processors used in the y direction
# npr0nam: 0 - parallel mpi I/O (mpi1 or mpi2)
```

```
#         1 - single mpi I/O (which uses a dedicated I/O processor for
#             MPI-1 only). When use this option, increase the number
#             of processors in your script by 1
#
# ianal:  analysis boundary condition switch
#          0 = compute output boundary tendencies for coarse domain only
#          1 = compute analysis and boundary tendencies
#          2 = compute analysis only
#          3 = compute nowcast only
# lm:     no. of atmospheric analysis pressure levels
# lmbc:   no. of atmospheric analysis boundary condition pressure levels
# nbdypt: number of grid points used for boundary condition blending
# ldigit: digital filter switch (logical)
# nbdya: boundary condition option used for digital filter
# idelay: forecast time (in hr, min, sec and one for each nest)
#         to start the delayed nest domain
# lcouple: coupled model switch (enables data structures and setup
#          necessary for coupled model run); this can set to false if one
#          is running atmos_forecast, otherwise it should be true;
#
# maxout: maximum number of ocards output
#
# iaero:  passive-tracer switch (0=off, 1=on)
# mspc:   number of tracer species
# nsrc:   number of emission sources for a single tracer
# mdust:  number of size bins in dust component (used when dustflg=t)
# lgrdall: passive tracer grid switch
#          t = passive tracer for all grids
#          f = passive tracer for specified grids only
# ntotcc: <=nnest, used if lgrdall=.true., specify total number of COAMPS
#         meshes to run aerosol-tracer (start with the coarse domain),
#         only work if all the nests are in different nest levels
# nestcc: used if lgrdall=.false., specify which grid nest to run
#         aerosol-tracer
#-----------------------------------------------------------------#
typeset nnestmx=7


#-----------------------------------------------------------------#
# number of atmospheric analysis pressure levels
# *** must be consistent with pr in coamnl setup ***
#-----------------------------------------------------------------#
typeset lm
case $atmos_analysis_type in
  3dvar) lm=${#host_atmos_prl[*]} ;;
  mvoi)  lm=16 ;;
esac
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
```

```
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
&atmosnl

 cdtg          = '${ddtg}',

 ndxnam        = ${nnestmx}*${atmos_nprocx},
 ndynam        = ${nnestmx}*${atmos_nprocy},
 npr0nam       = 0,

 ianal         = 1,
 lm            = ${lm},
 lmbc          = ${#host_atmos_prl[*]},
 ingres        = ${host_atmos_dimx},
 jngres        = ${host_atmos_dimy},
 nbdypt        = 7,
 ldigit        = f,
 nbdya         = 7,
 idelay        = 0, 0, 0,
                 0, 0, 0,
                 0, 0, 0,
                 0, 0, 0,
 lcouple       = t,

 maxout        = 12000,

 iaero         = 0,
 lgrdall       = f,
 mspc          = 1,
 nsrc          = 1,
 mdust         = 1,
 ntotcc        = 2,
 nestcc        = 1,

/
end_namelist
#-----------------------------------------------------------------#

}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-7 HOSTNL Namelist Setup Script (/projects/setup_nl_hostnl)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_hostnl $
# @(#)$Id: setup_nl_hostnl 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_hostnl
#
# Purpose:
#   Create the hostnl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_hostnl {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 \
site ddtg runDir gncomDir atmosDir obkgdDir \
|| return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: no input parameters for this function"; return 1
fi

#-----------------------------------------------------------------#
# namelist description:
#-----------------------------------------------------------------#
#       Name          Type                Description
#   -------------   --------    ----------------------------------
#   host_dsogrd     character   path to directory with host grid definition
#   host_dsodat     character   path to directory with host output files
#   host_run        character   host run identifier - e.g., 'none' or 'glb8_3c'
#   host_addyr      logical     if true, add the year to dsodat
#   host_patch2f    logical     it true, convert host to potential temp
#   host_updcyc     integer     update cycle for the host run (hours)
#   host_fcstbk     integer     lookback time for previous forecasts (hours)
#   host_fcstlen    integer     length of host forecast (default -1, ignored)
```

```
#                       When set > 0, it defines a minimum length in hours of the
#                       host forecast available and required. For boundary condition
#                       processing, if the setup cannot find a set of host files for
#                       a time past that minimum, the setup persists the last good
#                       set of data until the end of the NCOM forecast (by writing
#                       the same data labeled with a time at the end of the NCOM run).
#  host_tauinc    integer    frequency of host output (hours)
#  host_idtype    integer    type of host data
#                       1 - (G-, EAS-) NCOM output
#                       2 - RELO NCOM
#                       3 - NRL (G-,EAS-) netCDF format
#                       4 - NAVO netCDF format (not implemented)
#  host_nr        integer    number of tracers in the host output files
#  host_gclose    character  string type of index space closure of the host grid
#                       'none'  : No closure is applied.
#                       'smpl'  : Simple grid closure : Grid is periodic in the
#                             : i-index and wraps at i=NX+1. In other words,
#                             : (NX+1,J) => (1,J).
#                       'trpl'  : Tripole grid closure : Grid is periodic in the
#                             : i-index and wraps at i=NX+1 and has closure at
#                             : j=NY+1. In other words, (NX+1,J<=NY) => (1,J)
#                             : and (I,NY+1) => (MOD(NX-I+1,NX)+1,NY). Tripole
#                             : grid closure requires that NX be even.
#
#  htnd_gridnl    character  path to gridnl for host tendency fields
#  htnd_dsomsk    character  path to directory with land/sea mask flatfile
#                       for host tendency fields
#  htnd_dsotnd    character  path to directory for output of host tendency fields
#
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# setup
#-----------------------------------------------------------------#
typeset ymd=$(print $ddtg | cut -c1-8)

# host_fcst=t - use host-ocean forecast as lbc
#         =f - use host-ocean hindcast as lbc
# (this is not part of the namelist and is currently only used by the
#  retrieve from archive function)
typeset host_fcst=f

# site dependent settings
case $site in
#
# nrlssc:
#   - NetCDF format
#   - locally archived
#
  nrlssc)
```

```
   typeset host_idtype=2
   typeset host_run="none"
   typeset host_addyr=f
   typeset host_dsogrd="/u/COAMPS/scratch2/tasmith/Adr06valid/ocean"
   typeset host_dsodat="/u/COAMPS/scratch2/tasmith/Adr06valid/ocean"
   typeset host_gclose="none"
   ;;
#   typeset host_idtype=3
#   if (( $ymd < 20080601 ))
#   then
#     typeset host_run="glb8_2f"
#   else
#     typeset host_run="glb8_3b"
#   fi
#   typeset host_addyr=t
#   typeset host_dsogrd="/u/NCOM/glb8_2a/input"
#   typeset host_dsodat="/u/NCOM/${host_run}/nc"
#   typeset host_gclose="trpl"
#   ;;
#
# navy_dsrc:
#   - NetCDF format
#   - most recent forecast fields available in /scr/ooc/data/ncom1
#   - older fields assumed to be downloaded to $gncomDir
#
  navy_dsrc)
   typeset host_idtype=3
   if (( $ymd < 20080601 ))
   then
     typeset host_run="glb8_2f"
   else
     typeset host_run="glb8_3b"
   fi
   typeset host_addyr=f
   case $gncomDir in
     /scr/ooc/data/ncom1)
       typeset host_dsogrd="${gncomDir}/glb8_2a/input"
       typeset host_dsodat="${gncomDir}/${host_run}/nc"
       ;;
     *)
       typeset host_dsogrd="${gncomDir}"
       typeset host_dsodat="${gncomDir}"
       ;;
   esac
   typeset host_gclose="trpl"
   ;;
#
# other:
#   - native format (out3d)
#   - assumed to be downloaded to $gncomDir
#
```

```
  *)
    typeset host_idtype=1
    typeset host_run="none"
    typeset host_addyr=f
    typeset host_dsogrd="${gncomDir}"
    typeset host_dsodat="${gncomDir}"
    typeset host_gclose="trpl"
    ;;
esac
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
&hostnl

 host_dsogrd  = '${host_dsogrd}',
 host_dsodat  = '${host_dsodat}',

 host_idtype  = ${host_idtype},
 host_run     = '${host_run}',
 host_addyr   = ${host_addyr},

 host_patch2f = f,
 host_updcyc  = 12,
 host_fcstbk  = 12,
 host_fcstlen = 96,
 host_tauinc  = 1,
 host_nr      = 2,
 host_gclose  = '${host_gclose}',

 htnd_gridnl  = '${runDir}/gridnl.atmos',
 htnd_dsomsk  = '${atmosDir}',
 htnd_dsotnd  = '${obkgdDir}',

/
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-8 MVOI Namelist Script (/projects/setup_nl_mvoinl)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_mvoinl $
# @(#)$Id: setup_nl_mvoinl 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_mvoinl
#
# Purpose:
#   Create the mvoinl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_mvoinl {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 site ddtg atmosDir || return 1
case $site in
  fnmoc|coampsos) check_variables $0 aDir || return 1 ;;
  *)              check_variables $0 fandaDir || return 1 ;;
esac
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# namelist description:
#-----------------------------------------------------------------#
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
```

```
case $site in
fnmoc|coampsos)

cat << end_namelist
&mvoinl

 datfil = '${aDir}/a${ddtg}/',
 dsobsw = '${atmosDir}/',

/
end_namelist

;;

*)

cat << end_namelist
&mvoinl

 datfil = '${fandaDir}/adp${ddtg}/',
 dsobsw = '${atmosDir}/',

/
end_namelist

;;

esac
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-9 NCODA OANL Namelist Setup Script (/projects/setup_nl_oanl)

```
#-------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_oanl $
# @(#)$Id: setup_nl_oanl 407 2011-07-14 05:38:20Z campbell $
#-------------------------------------------------------------#
#
# Function: setup_nl_oanl
#
# Purpose:
#   Create the NCODA 3DVAR oanl namelist and write it to stdout.
#   The following choices of oanl namelists are possible:
#      ATMOS: oanl for 2D ocean analysis on the atmos grids
#      OCEAN: oanl for 3D ocean analysis on the ocean grids
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   1 (in) : type flag: "ATMOS" or "OCEAN"
#
#-------------------------------------------------------------#
function setup_nl_oanl {


#-------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------#
check_variables $0 update_cycle ocean_analysis_type || return 1
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------#
if [[ $# != 1 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
typeset tflag=$1
if [[ $tflag != "ATMOS" && $tflag != "OCEAN" ]]
then
  error_msg "$0: incorrect tflag input, choices are ATMOS or OCEAN"; return 1
fi
#-------------------------------------------------------------#

#-------------------------------------------------------------#
# namelist description: located in oceana/include/oanl.h
#-------------------------------------------------------------#
```

```
#-----------------------------------------------------------------#
# z_opt: vertical grid option
#       'none' - use vertical grid defined in z_lvl array
#       'ncom' - compute on sigma/z vertical grid
#                (z_lvl will be overridden with ncom z-levels obtained from
#                 file defined by NCOM_OZLVL_<nest>D)
#-----------------------------------------------------------------#
typeset z_opt=ncom


#-----------------------------------------------------------------#
# settings used when z_opt=none
#   n_lvl: number of analysis vertical grid levels
#   z_lvl: analysis vertical grid (meters)
#-----------------------------------------------------------------#
typeset n_lvl=30
typeset z_lvl_array=( \
    0.0    5.0   10.0   15.0   20.0   30.0 \
   50.0   75.0  100.0  125.0  150.0  200.0 \
  250.0  300.0  400.0  500.0  600.0  700.0 \
  800.0  900.0 1000.0 1100.0 1200.0 1300.0 \
 1400.0 1500.0 1750.0 2000.0 2500.0 3000.0 \
)
typeset z_lvl=$(print ${z_lvl_array[*]})
z_lvl=${z_lvl//+([[:space:]])/,}


#-----------------------------------------------------------------#
# BEGIN ATMOS NAMELIST
#-----------------------------------------------------------------#
if [[ $tflag == "ATMOS" ]]
then


case $ocean_analysis_type in
mvoi)
#-----------------------------------------------------------------#
# generate ATMOS namelist (MVOI)
#-----------------------------------------------------------------#
cat << end_namelist
&oanl

 upd_cyc  = ${update_cycle},
 locn3d   = 7*.false.,
 debug    = 10*.true.,

/
end_namelist
#-----------------------------------------------------------------#
;;
```

```
3dvar)
#-----------------------------------------------------------------#
# generate ATMOS namelist (3DVAR)
#-----------------------------------------------------------------#
cat << end_namelist
&oanl

 upd_cyc      = ${update_cycle},
 sst_asm      = .true.,

/
end_namelist
#-----------------------------------------------------------------#
;;
esac


return 0
fi
#-----------------------------------------------------------------#
# END ATMOS NAMELIST
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# BEGIN OCEAN NAMELIST
#-----------------------------------------------------------------#
if [[ $tflag == "OCEAN" ]]
then


case $ocean_analysis_type in
mvoi)
#-----------------------------------------------------------------#
# generate OCEAN namelist (MVOI)
#-----------------------------------------------------------------#
cat << end_namelist
&oanl

 debug    = 10*.true.,
 diurnal  = .true.,
 hc_mdl   = 'rsby',
 locn3d   = 7*.true.,
 lvl_nmo  = 1000.0,
 mask_opt = '3D',
 modas    = .true.,
 pool     = t,t,f,t,t,f,f,f,f,f,f
 prf_time = 'obst',
 pt_anl   = t,
 rscl     =   2,  2,  2,  2,  2
 ssh_std  = 2.0,
```

```
 ssh_time = 'obst',
 st_grd   = .true.,
 st_smpl  = 8,
 upd_cyc  = ${update_cycle},
 vc_bkg   = 'fcst',
 vc_mdl   = 'mixl',
 vol_scl =   6,  4,  6,  6,  8
 z_lvl    = ${z_lvl},

/
end_namelist
#-----------------------------------------------------------------#
;;
3dvar)
#-----------------------------------------------------------------#
# generate OCEAN namelist (3DVAR)
#-----------------------------------------------------------------#
cat << end_namelist
&oanl

 argo_bias    = .true.,
 bv_chk       = 2.0,
 debug(1)     = .true.,
 debug(2)     = .true.,
 debug(3)     = .true.,
 debug(5)     = .true.,
 debug(7)     = .true.,
 direct       = .false.,
 fcst(2)      = .true.,
 fcst(3)      = .false.,
 fcst(4)      = .true.,
 fgat         = -1, -1, -1, -1, -1,
 himem        = .true.,
 mask_opt     = '3D',
 modas        = .true.,
 model        = 'ncom',
 offset       = 10., 10., 10., 10.,
 n_pass       = 2,
 prf_slct(4)  = 20.
 prf_time     = 'obst',
 pt_anl       = .true.,
 rscl         = 2.0,1.2,1.2,1.2,2.0
 sal_asm      = .true.,
 ssh_asm      = .true.,
 ssh_mean     = 'data',
 ssh_time     = 'obst',
 sst_asm      = .true.,
 upd_cyc      = ${update_cycle},
 vc_mdl       = 'dens',
 z_lvl        = ${z_lvl},
 z_opt        = '${z_opt}',
```

```
/
end_namelist
#-----------------------------------------------------------------#
;;
esac


return 0
fi
#-----------------------------------------------------------------#
# END OCEAN NAMELIST
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-10 OMNL Namelist Setup Script (/projects/setup_nl_omnl)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_omnl $
# @(#)$Id: setup_nl_omnl 444 2011-10-21 20:40:54Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_omnl
#
# Purpose:
#   Create the omnl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_omnl {


#-----------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------------------------#
check_variables $0 gridnl_ocean || return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# namelist description:
#-----------------------------------------------------------------#
#
# inesta  : atmos nest to use for surface forcing
# dtcyc   : length of atmos update cycle (s)
# dtf     : time step for atmos surface forcing (s)
# dtt     : time step for atmos input sst (s)
#
#-----------------------------------------------------------------#
```

```
#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
&omnl

  inesta = 3,
  dtcyc  = 43200,
  dtf    = ${gridnl_ocean.nnest}*3600,
  dtt    = 3600,

/
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-11 OMNLOFF Namelist Setup Script (/projects/setup_nl_omnloff)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_omnloff $
# @(#)$Id: setup_nl_omnloff 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_omnloff
#
# Purpose:
#   Create the omnloff namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_omnloff {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 oceanDir || return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# namelist description:
#-----------------------------------------------------------------#
# outff   : output on (.true.) | off (.false.)
# out_dir : path to output files
#
# offpa   : surface atmospheric pressure
# offtx   : surface wind stress
# offep   : surface salt flux
# offqr   : solar radiation
# offq0   : surface heat flux
#
# offwb   : wave-bottom current (amp, dir, freq)
```

```
# offwr   : depth-integrated wave-radiation-stress gradient
# offsvs  : surface Stokes current (on staggered grid)
# offmvs  : 3D Stokes current (on native, staggered grid)
# offzvs  : 3D Stokes current (on z-levels, non-staggered)
#
# offse   : surface elevation
# offss   : surface salinity
# offst   : surface temperature
# offsv   : surface two-component (u,v) velocity (staggard)
#
# offms   : 3D salinity (on native grid)
# offmt   : 3D temperature (on native grid)
# offmv   : 3D velocity (on native, staggered grid)
#
# offzs   : 3D salinity on z-levels
# offzt   : 3D temperature on z-levels
# offzv   : 3D velocity on z-levels, non-staggered
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
&omnloff

 outff   = t,
 out_dir = '${oceanDir}/ffout',

 offpa   = 1.0,
 offtx   = 1.0,
 offep   = 1.0,
 offqr   = 1.0,
 offq0   = 1.0,

 offwb   = 1.0,
 offwr   = 1.0,
 offsvs  = 1.0,
 offmvs  = 1.0,
 offzvs  = 1.0,

 offse   = 1.0,
 offss   = 1.0,
 offst   = 1.0,
 offsv   = 1.0,

 offms   = 1.0,
 offmt   = 1.0,
 offmv   = 1.0,

 offzs   = 1.0,
```

```
 offzt   = 1.0,
 offzv   = 1.0,

/
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-12 SOILNL Namelist Setup Script (/projects/setup_nl_soilnl)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_soilnl $
# @(#)$Id: setup_nl_soilnl 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_nl_soilnl
#
# Purpose:
#   Create the soilnl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_nl_soilnl {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# namelist description:
#-----------------------------------------------------------------#
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
&soilnl
```

```
 nsoil  =    4,
 iagrmet=    0,

/
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-13 Post Processing Script (/projects/setup_nl_postnl)

```
#-------------------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_nl_postnl $
# @(#)$Id: setup_nl_postnl 415 2011-07-21 20:40:31Z campbell $
#-------------------------------------------------------------------------------#
#
# Function: setup_nl_postnl
#
# Purpose:
#   Create the postnl namelist and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-------------------------------------------------------------------------------#
function setup_nl_postnl {


#-------------------------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------------------------#
check_variables $0 oceanDir ddtg fcst_length || return 1
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# namelist description:
#-------------------------------------------------------------------------------#
#   post_distrib    character   flag for distribution statement in output netCDF
#       'public':
#           'Approved for public release; distribution is unlimited.'
#       'navo_restrict':
#           'Distribution limited to Department of Defense and U.S. DOD '
#           'contractors only.  Administrative/operational <today_date>.'
#           'Other U.S. requests must be referred to the Commanding '
#           'Officer, Naval Oceanographic Office.'
#       'fnmoc_restrict':
#           'Distribution limited to Department of Defense and U.S. DOD '
```

```
#          'contractors only.  Administrative/operational <today_date>.'
#          'Other U.S. requests must be referred to the Commanding '
#          'Officer, Fleet Numerical Meteorology and Oceanography Center.'
#       otherwise, the string in post_distrib is written as is
#
#  post_dsodat     character   path to directory with NCOM output flatfiles
#     this variable is set at run time by the scripts
#  post_outdir     character   path to directory for NCOM NetCDF output
#  post_dtganal    character   10-character nowcast or analysis time
#  post_dtgepoch   character   10-character reference time for time() variable
#  post_hcstlen    integer     length of NCOM hindcast to process (hours)
#  post_fcstlen    integer     length of NCOM forecast to process (hours)
#  post_tauinc     integer     frequency of NCOM output to process (hours)
#     must be a multiple of the required off* flatfile output intervals
#  post_taufile    logical     individual NetCDF output file for each tau
#
#  post_outind     logical     vector of indices to turn on postprocessing
#     default: off
#     (1)  surface elevation (requires offse)
#     (2)  surface temperature (requires offst)
#     (3)  surface salinity (requires offss)
#     (4)  surface velocity (requires offsv)
#     (5)  velocity (requires offmv)
#     (6)  temperature (requires offmt)
#     (7)  salinity (requires offms)
#     (8)  bottom currents in shallow water (requires offmv)
#     (9)  surface forcing fields (requires one or more of
#          offpa, offtx, offep, offqr, offq0)
#     (10) potential temperature (requires offmt)
#     (11) soundspeed (requires offmt+offms or offzt+offzs)
#     (12) density anomaly
#
#  post_ivflag     integer     use linear (1) or PCHIP (2) profile interp
#  post_appgrd     real        bounds and spacing of output application grid
#     (lonmin, lonmax, latmin, latmax, dlon, dlat) all in degrees
#     the default output application grid is a minimal lat/lon bounding box
#     of the computational grid with grid spacing based on mean spacing of
#     the computational grid
#     the default is obtained by setting any value in post_appgrd to -999
#  post_nlev       integer     number of level depths for the NetCDF output
#  post_zlev       real        level depths for the NetCDF output (meters)
#-------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------#
# generate namelist
#-------------------------------------------------------------------------------#
cat << end_namelist
&postnl

 post_distrib    = 'none',
```

```
 post_outdir     = '${oceanDir}/ncout',
 post_dtganal    = '${ddtg}',
 post_hcstlen    = 0,
 post_fcstlen    = ${fcst_length},
 post_tauinc     = 3,
 post_taufile    = t,

 post_outind     = t,t,t,t,t,t,t,t,t,t,t,t,

 post_ivflag     = 2,
 post_appgrd     = 6*-999,
 post_nlev       = 34,
 post_zlev       =    0.0,    2.5,    7.5,   12.5,   18.0,   25.0,   32.5,
                     40.0,   50.0,   62.0,   75.0,  100.0,  125.0,  150.0,
                    200.0,  300.0,  400.0,  500.0,  600.0,  700.0,  800.0,
                    900.0, 1000.0, 1100.0, 1200.0, 1300.0, 1400.0, 1500.0,
                   1750.0, 2000.0, 2500.0, 3000.0, 4000.0, 5000.0,

/
end_namelist
#-----------------------------------------------------------------------------#


}
#-----------------------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------------------#
```

B-14 SWAN Input Data Script (/projects/setup_swan_input)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_swan_input $
# @(#)$Id: setup_swan_input 472 2011-11-30 14:39:34Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_swan_input
#
# Purpose:
#   Create the SWAN input data and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   1 (in) : mode (coupled|standalone)
#
#-----------------------------------------------------------------#
function setup_swan_input {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 \
newdtg ddtg update_cycle fcst_length area wave_nprocs waveDir prjDir \
gridnl_wave couple_a2w couple_o2w couple_w2o atmos_enabled ocean_enabled \
|| return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 1 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
typeset mode=$1
case $mode in
  coupled|standalone) ;;
  *) error_msg "$0: unsupported mode"; return 1 ;;
esac
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# make sure newdtg exists and is executable
#-----------------------------------------------------------------#
```

```
if [[ ! -x $newdtg ]]
then
  error_msg "$0: $newdtg does not exist or does not have execute permission"
  return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# timestep in seconds
#-----------------------------------------------------------------#
typeset -i dtw=60
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# start day and hour/minute of run
#-----------------------------------------------------------------#
typeset start_day=$(print $ddtg | cut -c1-8)
typeset start_hrm=$(print $ddtg | cut -c9-10)00
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# end day and hour/minute of run
#-----------------------------------------------------------------#
typeset edtg=$(print $ddtg \ +$fcst_length | $newdtg 2>/dev/null)
typeset end_day=$(print $edtg | cut -c1-8)
typeset end_hrm=$(print $edtg | cut -c9-10)00
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# restart output day and hour/minute of run
#-----------------------------------------------------------------#
typeset ndtg=$(print $ddtg \ +$update_cycle | $newdtg 2>/dev/null)
typeset rstrt_day=$(print $ndtg | cut -c1-8)
typeset rstrt_hrm=$(print $ndtg | cut -c9-10)00
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# define output fields
#-----------------------------------------------------------------#
typeset output_fields="
output_fields="HSIGN HSWELL DIR PDIR RTP TM01 TM02 WLEN"
output_fields="$output_fields WIND TAUW UFRI"
output_fields="$output_fields VEL WATLEV"
output_fields="$output_fields FORCE UBOT URMS TMBOT"
#-----------------------------------------------------------------#
```

```
#-----------------------------------------------------------------#
# spatial grid defined by gridnl
#-----------------------------------------------------------------#
typeset -i mcx=$(( ${gridnl_wave.m[1]} - 1 ))
typeset -i mcy=$(( ${gridnl_wave.n[1]} - 1 ))
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# set initial conditions
#-----------------------------------------------------------------#
typeset initial_conditions="
if [[ $(get_wave_restart_flag) == [tT] ]]
then
  if [[ $wave_restart_type == multi ]]
  then
    initial_conditions="INITIAL HOTSTART MULTIPLE 'rstrt_${ddtg}'"
  else
    initial_conditions="INITIAL HOTSTART SINGLE 'rstrt_${ddtg}'"
  fi
else
  initial_conditions="INITIAL DEFAULT"
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# set boundary conditions
#-----------------------------------------------------------------#
typeset boundary_conditions="
#boundary_conditions="BOUN NEST '${area}_${ddtg}' CLOSED"
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# set surface forcing (file based)
#-----------------------------------------------------------------#
typeset wind_inpgrid="
typeset wind_readinp="
if [[ $atmos_enabled == [fF] ]]
then
  wind_inpgrid="INPGRID WIND CURV 0 0 ${mcx} ${mcy} NONSTAT
${start_day}.${start_hrm} 1 HR"
  wind_readinp="READINP WIND 1.0 'wind_${ddtg}.dat' 4 UNFORMATTED"
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# set ocean forcing (file based)
```

```
#-----------------------------------------------------------------#
typeset wlev_inpgrid="
typeset wlev_readinp="
if [[ $ocean_enabled == [fF] ]]
then
  wlev_inpgrid="INPGRID WLEV CURV 0 0 ${mcx} ${mcy} NONSTAT
${start_day}.${start_hrm} 1 HR"
  wlev_readinp="READINP WLEV 1.0 'wlev_${ddtg}.dat' 4 UNFORMATTED"
fi
typeset cvel_inpgrid="
typeset cvel_readinp="
if [[ $ocean_enabled == [fF] ]]
then
  cvel_inpgrid="INPGRID CURR CURV 0 0 ${mcx} ${mcy} NONSTAT
${start_day}.${start_hrm} 1 HR"
  cvel_readinp="READINP CURR 1.0 'cvel_${ddtg}.dat' 4 UNFORMATTED"
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate non-mode dependent part of input
#-----------------------------------------------------------------#
cat << end_namelist
$ **************HEADING*************************************
$
PROJ '${area}' '----'
$
$ ************MODEL INPUT************************************
$
COORDINATES SPHERICAL
$
$ Computational grid
CGRID CURV ${mcx} ${mcy} EXCEPTION 999 999 CIRCLE 36 0.0418 1.0 33
READGRID COORD 1.0 'grid.dat' 4 UNFORMATTED
$
$ Input grids
INPGRID BOTTOM CURV EXCEPTION 0
READINP BOTTOM 1.0 'bottom.dat' 4 UNFORMATTED
$
$ Initial conditions
${initial_conditions}
$
$ Boundary conditions
${boundary_conditions}
$
$ Surface forcing (file based)
${wind_inpgrid}
${wind_readinp}
$
$ Ocean forcing (file based)
```

```
${wlev_inpgrid}
${wlev_readinp}
${cvel_inpgrid}
${cvel_readinp}
$
$ Source/sink terms and conv. criterion
$ *** Old Physics ***
$ GEN3 KOMEN AGROW
$  WCAP KOM 2.36E-5 3.02E-3 2.0 1.0 1.0
$ *** New Physics ***
 GEN3 BABANIN 5.7E-7 8.0E-6 4.0 4.0 1.2 0.0060 UP AGROW
$ ********************
FRIC JON 0.019
QUAD iquad=8
$
$ Numerics, etc.
NUM ACCUR 0.02 0.02 0.02 98.0 STAT 50
NUM ACCUR 0.02 0.02 0.02 98.0 NONST 1
 PROP GSE 1.8 HR
$
$
$ ***********OUTPUT REQUESTS**********************************
$
$ *** Matlab Output ***
BLOCK 'COMPGRID' NOHEAD 'grid_${ddtg}.mat' LAY 3 &
  DEPTH XP YP OUTPUT ${start_day} 100 DAY
BLOCK 'COMPGRID' NOHEAD 'output_${ddtg}.mat' LAY 3 &
  ${output_fields} &
  OUTPUT ${start_day} 1 HR
$
$ *** ASCII Output ***
$ BLOCK 'COMPGRID' HEAD 'grid_${ddtg}' LAY 4 &
$   DEPTH XP YP OUTPUT ${start_day} 100 DAY
$ BLOCK 'COMPGRID' HEAD 'output_${ddtg}' LAY 4 &
$   ${output_fields} &
$   OUTPUT ${start_day} 1 HR
$
$ *** 1D Spectra Output ***
$ SPECOUT 'COMPGRID' SPEC1D ABSOLUTE 'spec1d_${ddtg}' &
$   OUTPUT ${start_day} 1 HR
$
$ *** 2D Spectra Output ***
$ SPECOUT 'COMPGRID' SPEC2D ABSOLUTE 'spec2d_${ddtg}' &
$   OUTPUT ${start_day} 3 HR
$
$ *** Nest Output ***
$ NONE
$
$
$ ***********COMPUTE*****************************************
$
```

```
end_namelist
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate mode dependent part of input
#-----------------------------------------------------------------#
case $mode in

coupled)
cat << end_namelist
HOTFILE 'rstrt_${ndtg}' ${update_cycle} HR SNGLOUT
COMPUTE ${dtw} SEC
STOP
end_namelist
;;

standalone)
if [[ $(get_wave_restart_flag) == [fF] ]]
then
cat << end_namelist
COMPUTE STAT  ${start_day}.${start_hrm}
end_namelist
fi
if (( $fcst_length > $update_cycle ))
then
cat << end_namelist
COMPUTE NONST ${start_day}.${start_hrm} ${dtw} SEC ${rstrt_day}.${rstrt_hrm}
HOTFILE 'rstrt_${ndtg}'
COMPUTE NONST ${rstrt_day}.${rstrt_hrm} ${dtw} SEC ${end_day}.${end_hrm}
STOP
end_namelist
else
cat << end_namelist
COMPUTE NONST ${start_day}.${start_hrm} ${dtw} SEC ${end_day}.${end_hrm}
STOP
end_namelist
fi
;;

esac
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-15 WAVEWATCH III Initialization Script (setup_ww3_strt)

```
#-------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_ww3_strt $
# @(#)$Id: setup_ww3_strt 407 2011-07-14 05:38:20Z campbell $
#-------------------------------------------------------------#
#
# Function: setup_ww3_strt
#
# Purpose:
#   Create the ww3_strt input and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-------------------------------------------------------------#
function setup_ww3_strt {


#-------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------#
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# generate namelist
#-------------------------------------------------------------#
cat << end_namelist
$ ---------------------------------------------------------- $
$ WAVEWATCH III Initial conditions input file                  $
$ ---------------------------------------------------------- $
$ type of initial field ITYPE .
$
  3
$
$ ITYPE = 1 ------------------------------------------------- $
$ Gaussian in frequency and space, cos type in direction.
$ - fp and spread (Hz), mean direction (degr., oceanographic
```

```
$    convention) and cosine power, Xm and spread (degr. or m) Ym and
$    spread (degr. or m), Hmax (m) (Example for lon-lat grid in degr.).
$
$   0.10  0.01  270. 2  239.0 0.2 45.0 0.2 2.5
$
$ ITYPE = 2 ---------------------------------------------------- $
$ JONSWAP spectrum with Hasselmann et al. (1980) direct. distribution.
$ - alfa, peak freq. (Hz), mean direction (degr., oceanographical
$   convention), gamma, sigA, sigB, Xm and spread (degr. or m) Ym and
$   spread (degr. or m)  (Example for lon-lat grid in degr.).
$   alfa, sigA, sigB give default values if less than or equal to 0.
$
$   0.0081  0.1  270.  1.0 0. 0. 1. 100. 1. 100.
$
$ ITYPE = 3 ---------------------------------------------------- $
$ Fetch-limited JONSWAP
$ - No additional data, the local spectrum is calculated using the
$   local wind speed and direction, using the spatial grid size as
$   fetch, and assuring that the spectrum is within the discrete
$   frequency range.
$
$ ITYPE = 4 ---------------------------------------------------- $
$ User-defined spectrum
$ - Scale factor., defaults to 1 if less than or equal 0.
$ - Spectrum F(f,theta) (single read statement)
$
$ -0.1
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 1 4 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 1 2 3 2 1 1 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 1 3 9 7 5 3 2 1 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 1 3 4 3 2 1 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
$
$ ITYPE = 5 ------------------------------------------------- $
$ Starting from calm conditions.
$ - No additional data.
$
$ ------------------------------------------------------------ $
$ End of input file                                          $
$ ------------------------------------------------------------ $
end_namelist
#-------------------------------------------------------------#


}
#-------------------------------------------------------------#
# end of function
#-------------------------------------------------------------#
```

B-16 WAVEWATCH III Grid Setup Script (setup_ww3_grid)

```
#-------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_ww3_grid $
# @(#)$Id: setup_ww3_grid 407 2011-07-14 05:38:20Z campbell $
#-------------------------------------------------------------#
#
# Function: setup_ww3_grid
#
# Purpose:
#   Create the ww3_grid input and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-------------------------------------------------------------#
function setup_ww3_grid {


#-------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------------------#
check_variables $0 \
area gridnl_wave \
|| return 1
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# timestep in seconds
#-------------------------------------------------------------#
typeset -i dtw=180
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# spatial grid defined by gridnl
```

```
#-------------------------------------------------------------#
typeset -i nx=$(( ${gridnl_wave.m[1]} ))
typeset -i ny=$(( ${gridnl_wave.n[1]} ))
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# generate namelist
#-------------------------------------------------------------#
cat << end_namelist
$ ------------------------------------------------------------- $
$ WAVEWATCH III Grid preprocessor input file                     $
$ ------------------------------------------------------------- $
$ Grid name (C*30, in quotes)
$
  '${area}'
$
$ Frequency increment factor and first frequency (Hz) ----------- $
$ number of frequencies (wavenumbers) and directions, relative offset
$ of first direction in terms of the directional increment [-0.5,0.5].
$ In versions 1.18 and 2.22 of the model this value was by definiton 0,
$ it is added to mitigate the GSE for a first order scheme. Note that
$ this factor is IGNORED in the print plots in ww3_outp.
$
  1.1 0.0418  33  36  0.
$
$ Set model flags ---------------------------------------------- $
$  - FLDRY         Dry run (input/output only, no calculation).
$  - FLCX, FLCY    Activate X and Y component of propagation.
$  - FLCTH, FLCK   Activate direction and wavenumber shifts.
$  - FLSOU         Activate source terms.
$
  F T T T T T
$
$ Set time steps ----------------------------------------------- $
$ - Time step information (this information is always read)
$     maximum global time step, maximum CFL time step for x-y and
$     k-theta, minimum source term time step (all in seconds).
$
  ${dtw} 180. 180. 60.
$
$ Start of namelist input section ------------------------------- $
$   Starting with WAVEWATCH III version 2.00, the tunable parameters
$   for source terms, propagation schemes, and numerics are read using
$   namelists. Any namelist found in the folowing sections up to the
$   end-of-section identifier string (see below) is temporarily written
$   to ww3_grid.scratch, and read from there if necessary. Namelists
$   not needed for the given switch settings will be skipped
$   automatically, and the order of the namelists is immaterial.
$
  &PRO3 WDTHCG = 1.50, WDTHTH = 1.50 /
```

```
$
$ Mandatory string to identify end of namelist input section.
$
END OF NAMELISTS
$
$ Define grid ------------------------------------------------- $
$
$ Five records containing :
$
$  1 Type of grid, coordinate system and global flag: GSTRG, FLAGLL,
$    GLOBAL.  A global grid may be rectilinear or curvilinear.
$    A global grid must be in spherical coordinates.
$      GSTRG  : String indicating type of grid :
$               'RECT'  : rectilinear
$               'CURV'  : curvilinear
$      FLAGLL : Flag to indicate coordinate system :
$               T  : Spherical (lon/lat in degrees)
$               F  : Cartesian (meters)
$      GLOBAL : Flag to indicate whether or not the grid is global :
$               T  : Global (longitudinal closure is applied)
$               F  : Not global
$  2 NX, NY. As the outer grid lines are always defined as land
$    points, the minimum size is 3x3.
$
$  3 Unit number of file with x-coordinate.
$    Scale factor and add offset: x <= scale_fac * x_read + add_offset.
$    IDLA, IDFM, format for formatted read, FROM and filename.
$      IDLA : Layout indicator :
$                1   : Read line-by-line bottom to top.
$                2   : Like 1, single read statement.
$                3   : Read line-by-line top to bottom.
$                4   : Like 3, single read statement.
$      IDFM : format indicator :
$                1   : Free format.
$                2   : Fixed format with above format descriptor.
$                3   : Unformatted.
$      FROM : file type parameter
$               'UNIT' : open file by unit number only.
$               'NAME' : open file by name and assign to unit.
$
$    If the above unit number equals 10, then the x-coord is read from this
$    file.  The x-coord must follow the above record.  No comment lines are
$    allowed within the x-coord input.
$
$  4 Unit number of file with y-coordinate.
$    Scale factor and add offset: y <= scale_fac * y_read + add_offset.
$    IDLA, IDFM, format for formatted read, FROM and filename.
$      IDLA : Layout indicator :
$                1   : Read line-by-line bottom to top.
$                2   : Like 1, single read statement.
$                3   : Read line-by-line top to bottom.
```

```
$              4   : Like 3, single read statement.
$     IDFM : format indicator :
$              1   : Free format.
$              2   : Fixed format with above format descriptor.
$              3   : Unformatted.
$     FROM : file type parameter
$            'UNIT' : open file by unit number only.
$            'NAME' : open file by name and assign to unit.
$
$   If the above unit number equals 10, then the y-coord is read from this
$   file.  The y-coord must follow the above record.  No comment lines are
$   allowed within the y-coord input.
$
$  5 Limiting bottom depth (m) to discriminate between land and sea
$   points, minimum water depth (m) as allowed in model, unit number
$   of file with bottom depths, scale factor for bottom depths (mult.),
$   IDLA, IDFM, format for formatted read, FROM and filename.
$     IDLA : Layout indicator :
$              1   : Read line-by-line bottom to top.
$              2   : Like 1, single read statement.
$              3   : Read line-by-line top to bottom.
$              4   : Like 3, single read statement.
$     IDFM : format indicator :
$              1   : Free format.
$              2   : Fixed format with above format descriptor.
$              3   : Unformatted.
$     FROM : file type parameter
$            'UNIT' : open file by unit number only.
$            'NAME' : open file by name and assign to unit.
$
$   If the above unit number equals 10, then the bottom depths are read from
$   this file.  The depths must follow the above record.  No comment lines are
$   allowed within the depth input.
$
 'CURV' T F
 ${nx} ${ny}
 20 1.0 0.0 2 3 '(....)' 'NAME' 'gridx.dat'
 21 1.0 0.0 2 3 '(....)' 'NAME' 'gridy.dat'
 -0.1 5.0 22 -1.0 2 3 '(....)' 'NAME' 'bottom.dat'
$
$ Input boundary points and excluded points ---------------------- $
$
 10 3 1 '(....)' 'PART' 'mapsta.inp'
$
$ Read the status map from file ( FROM != PART ) ---------------- $
$
$
$ The legend for the input map is :
$
$   0 : Land point.
$   1 : Regular sea point.
```

```
$    2 : Active boundary point.
$    3 : Point excluded from grid.
$
$ Input boundary points from segment data ( FROM = PART ) ------- $
$   An unlimited number of lines identifying points at which input
$   boundary conditions are to be defined. If the actual input data is
$   not defined in the actual wave model run, the initial conditions
$   will be applied as constant boundary conditions. Each line contains:
$     Discrete grid counters (IX,IY) of the active point and a
$     connect flag. If this flag is true, and the present and previous
$     point are on a grid line or diagonal, all intermediate points
$     are also defined as boundary points.
$
    0   0   F
$
$ Excluded grid points from segment data ( FROM != PART )
$   First defined as lines, identical to the definition of the input
$   boundary points, and closed the same way.
$
    0   0   F
$
$   Second, define a point in a closed body of sea points to remove
$   the entire body of sea points. Also close by point (0,0)
$
    0   0
$
$ Output boundary points ---------------------------------------- $
$ Output boundary points are defined as a number of straight lines,
$ defined by its starting point (X0,Y0), increments (DX,DY) and number
$ of points. A negative number of points starts a new output file.
$ Note that this data is only generated if requested by the actual
$ program. Example again for spherical grid in degrees. Note, these do
$ not need to be defined for data transfer between grids in te multi
$ grid driver.
$
   0.    0.    0.    0.       0
$
$ ---------------------------------------------------------------- $
$ End of input file                                          $
$ ---------------------------------------------------------------- $
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

B-17 WAVEWATCH III Multi Input Setup Script (setup_ww3_multi)

```
#------------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_ww3_multi $
# @(#)$Id: setup_ww3_multi 407 2011-07-14 05:38:20Z campbell $
#------------------------------------------------------------------#
#
# Function: setup_ww3_multi
#
# Purpose:
#   Create the ww3_multi input and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#------------------------------------------------------------------#
function setup_ww3_multi {


#------------------------------------------------------------------#
# check for required global variables
#------------------------------------------------------------------#
check_variables $0 \
area newdtg ddtg update_cycle fcst_length couple_a2w couple_o2w \
|| return 1
#------------------------------------------------------------------#


#------------------------------------------------------------------#
# process input parameters
#------------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#------------------------------------------------------------------#


#------------------------------------------------------------------#
# make sure newdtg exists and is executable
#------------------------------------------------------------------#
if [[ ! -x $newdtg ]]
then
  error_msg "$0: $newdtg does not exist or does not have execute permission"
  exit 1
fi
#------------------------------------------------------------------#
```

```
#-----------------------------------------------------------------#
# define date-time of run
#-----------------------------------------------------------------#
# start date-time
typeset sdate=$(print $ddtg | cut -c1-8)
typeset stime=$(print $ddtg | cut -c9-10)0000
# end date-time
typeset edtg=$(print $ddtg \ +$fcst_length | $newdtg 2>/dev/null)
typeset edate=$(print $edtg | cut -c1-8)
typeset etime=$(print $edtg | cut -c9-10)0000
# update cycle in seconds
typeset upsec=$(( update_cycle * 3600 ))
# output intervals in seconds
typeset doutf=3600
typeset doutp=3600
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# define input forcing
#-----------------------------------------------------------------#
if [[ $couple_a2w == [tT] ]]
then
  wind_inp="coupler"
else
  wind_inp="no"
fi
if [[ $couple_o2w == [tT] ]]
then
  wcur_inp="coupler"
  wlev_inp="coupler"
else
  wcur_inp="no"
  wlev_inp="no"
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# generate namelist
#-----------------------------------------------------------------#
cat << end_namelist
$ ------------------------------------------------------------- $
$ WAVEWATCH III multi-grid model driver input file                $
$ ------------------------------------------------------------- $
$
$ The first input line sets up the general multi-grid model definition
$ by defining the follwing six parameters :
$
$   1) Number of wave model grids.i                    ( NRGRD )
$   2) Number of grids definint input fields.           ( NRINP )
```

```
$   3) Flag for using unified point output file.          ( UNIPTS )
$   4) Output server type as in ww3_shel.inp
$   5) Flag for dedicated process for iunified point output.
$   6) Flag for grids sharing dedicated output processes.
$
  1 0 F 1 F F
$
$ -------------------------------------------------------------- $
$ If there are input data grids defined ( NRINP > 0 ), then these
$ grids are defined first. These grids are defined as if they are wave
$ model grids using the file mod_def.MODID. Each grid is defined on
$ a separate input line with MODID, and eight input flags identifying
$ the presentce of 1) water levels 2) currents 3) winds 4) ice and
$ 5-7) assimilation data as in the file ww3_shel.inp.
$
$  'ww3' F F T F F F F
$
$ In this example, we need the file mod_def.ww3 to define the grid
$ and the file wind.ww3 to provide the corresponding wind data.
$
$ -------------------------------------------------------------- $
$ If all point output is gathered in a unified point output file
$ ( UNIPTS = .TRUE. ), then the output spectral grid needs to be
$ defined. Ths information is taken from a wave model grid, and only
$ the spectral definitions from this grid are relevant. Define the
$ name of this grid here
$
$  'points'
$
$ In this example, we need the file mod_def.points to define the
$ spectral output grid, and the point output will be written to the
$ file out_pnt.points
$
$ -------------------------------------------------------------- $
$ Now each actual wave model grid is defined using 13 parameters to be
$ read fom a single line in the file. Each line contains the following
$ parameters
$     1)   Define the grid with the extension of the mod_def file.
$    2-8)  Define the inputs used by the grids with 8 keywords
$          corresponding to the 8 flags defining the input in the
$          input files. Valid keywords are:
$            'no'      : This input is not used.
$            'native'  : This grid has its own input files, e.g. grid
$                        grdX (mod_def.grdX) uses ice.grdX.
$            'coupler' : Take input from a coupled model.
$            'MODID'   : Take input from the grid identified by
$                        MODID. In the example below, all grids get
$                        their wind from wind.input (mod_def.input).
$     9)   Rank number of grid (internally sorted and reassigned).
$    10)   Group number (internally reassigned so that different
$          ranks result in different group numbers.
```

```
$   11-12) Define fraction of cumminicator (processes) used for this
$          grid.
$    13)   Flag identifying dumping of boundary data used by this
$          grid. If true, the file nest.MODID is generated.
$ -------------------------------------------------------------- $
  'ww3' '${wlev_inp}' '${wcur_inp}' '${wind_inp}' 'no' 'no' 'no' 'no'   1  1  0.00 1.00  F
$
$ -------------------------------------------------------------- $
$ Starting and ending times for the entire model run
$
  ${sdate} ${stime} ${edate} ${etime}
$
$ -------------------------------------------------------------- $
$ Specific multi-scale model settings (single line).
$    Flag for masking computation in two-way nesting (except at
$                                        output times).
$    Flag for masking at printout time.
$
  T  T
$
$ -------------------------------------------------------------- $
$ Conventional output requests as in ww3_shel.inp.  Will be applied
$ to all grids.
$
  ${sdate} ${stime}  ${doutf}  ${edate} ${etime}
  T T T F F  T T T T F  T T F F T  T F T F F  F F F F F  F F F F F  F
$
$ Type 2 : Point output
  ${sdate} ${stime}  ${doutp}  ${edate} ${etime}
  0.0   0.0  'STOPSTRING'
$
$ Type 3 : Output along  track.
  ${sdate} ${stime}  0000  ${edate} ${etime}
$  T
$
$ Type 4 : Restart files
  ${sdate} ${stime}  ${upsec}  ${edate} ${etime}
$
$ Type 5 : Boundary data
  ${sdate} ${stime}  0000  ${edate} ${etime}
$
$ Type 6 : Separated wave field data (dummy for now).
$          First, last step IX and IY, flag for formatted file
  ${sdate} ${stime}  0000  ${edate} ${etime}
$
$ -------------------------------------------------------------- $
$ Output requests per grid and type
$
$  'grd3'  6
$  ${sdate} ${stime}  0000  ${edate} ${etime}
$  0 999 1 0 999 1 T
```

```
$
$ ---------------------------------------------------------- $
$ Mandatory end of outpout requests per grid, identified by output
$ type set to 0.
$
  'the_end'  0
$
$ ---------------------------------------------------------- $
$ Moving grid data as in ww3_hel.inp. All grids will use same data.
$
$  'MOV' ${sdate} ${stime}   5.  90.
  'STP'
$
$ ---------------------------------------------------------- $
$ End of input file                                          $
$ ---------------------------------------------------------- $
end_namelist
#-------------------------------------------------------------#


}
#-------------------------------------------------------------------#
# end of function
#-------------------------------------------------------------------#
```

B-18 WAVEWATCH III Outf Input Setup Script (setup_ww3_outf)

```
#-----------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_ww3_outf $
# @(#)$Id: setup_ww3_outf 407 2011-07-14 05:38:20Z campbell $
#-----------------------------------------------------------------#
#
# Function: setup_ww3_outf
#
# Purpose:
#   Create the ww3_outf input and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-----------------------------------------------------------------#
function setup_ww3_outf {


#-----------------------------------------------------------------#
# check for required global variables
#-----------------------------------------------------------------#
check_variables $0 \
area prjDir ddtg update_cycle fcst_length \
|| return 1
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# process input parameters
#-----------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
# start date-time of run
#-----------------------------------------------------------------#
typeset sdate=$(print $ddtg | cut -c1-8)
typeset stime=$(print $ddtg | cut -c9-10)0000
#-----------------------------------------------------------------#


#-----------------------------------------------------------------#
```

```
# output interval and number of outputs
#--------------------------------------------------------------#
typeset dout=$(shvar doutf $prjDir/setup_ww3_multi) \
  || { error_msg "$0:$LINENO: get ww3 doutf failed"; return 1; }
typeset nout=$(( 1 + fcst_length * 3600 / dout ))
#--------------------------------------------------------------#


#--------------------------------------------------------------#
# generate namelist
#--------------------------------------------------------------#
cat << end_namelist
$ ------------------------------------------------------------ $
$ WAVEWATCH III Grid output post-processing                    $
$ ------------------------------------------------------------ $
$ Time, time increment and number of outputs
$
  ${sdate} ${stime} ${dout} ${nout}
$
$ Request flags identifying fields as in ww3_shel input and
$ section 2.4 of the manual.
$
  T T T F F  T T T T F  T T F F T  T F T F F  F F F F F  F F F F F  F
$
$ Output type ITYPE [0,1,2,3], and IPART [ 0,...,NOSWLL ]
$
  3 0
$
$ ------------------------------------------------------------ $
$ ITYPE = 0, inventory of file.
$          No additional input, the above time range is ignored.
$
$ ------------------------------------------------------------ $
$ ITYPE = 1, print plots.
$          IX,IY range and stride, flag for automatic scaling to
$          maximum value (otherwise fixed scaling),
$          vector component flag (dummy for scalar quantities),
$
$ 1 10 1 1 10 1 F F
$
$ ------------------------------------------------------------ $
$ ITYPE = 2, field statistics.
$          IX,IY range.
$
$ 1 10 1 10
$
$ ------------------------------------------------------------ $
$ ITYPE = 3, transfer files.
$          IX, IY range, IDLA and IDFM as in ww3_grid.inp.
$          The additional option IDLA=5 gives ia longitude, lattitude
$          and parameter value(s) per record (defined points only),
```

```
$
  1 9999 1 9999 1 1
$
$ For each field and time a new file is generated with the file name
$ ww3.yymmddhh.xxx, where yymmddhh is a conventional time idicator,
$ and xxx is a field identifier. The first record of the file contains
$ a file ID (C*13), the time in yyyymmdd hhmmss format, the lowest,
$ highest and number of longitudes (2R,I), id.  latitudes, the file
$ extension name (C*$), a scale factor (R), a unit identifier (C*10),
$ IDLA, IDFM, a format (C*11) and a number identifying undefined or
$ missing values (land, ice, etc.).  The field follows as defined by
$ IDFM and IDLA, defined as in the grid proprocessor. IDLA=5 is added
$ and gives a set of records containing the longitude, latitude and
$ parameter value. Note that the actual data is written as an integers.
$
$ ------------------------------------------------------------ $
$ End of input file                                            $
$ ------------------------------------------------------------ $
end_namelist
#-------------------------------------------------------------#


}
#-------------------------------------------------------------#
# end of function
#-------------------------------------------------------------#
```

B-19 WAVEWATCH III Outp Input Setup Script (setup_ww3_outp)

```
#-------------------------------------------------------------#
# CONFIGURATION IDENTIFICATION:
# $HeadURL: http://coamps.nrlmry.navy.mil:8000/svn/run-
coamps5/branches/ssc/projects/socal/setup_ww3_outp $
# @(#)$Id: setup_ww3_outp 407 2011-07-14 05:38:20Z campbell $
#-------------------------------------------------------------#
#
# Function: setup_ww3_outp
#
# Purpose:
#   Create the ww3_outp input and write it to stdout.
#   If an error occurs, then a non-zero exit status is returned.
#
# Required calling parameters:
#   NONE
#
#-------------------------------------------------------------#
function setup_ww3_outp {


#-------------------------------------------------------------#
# check for required global variables
#-------------------------------------------------------------#
check_variables $0 \
area prjDir ddtg update_cycle fcst_length \
|| return 1
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# process input parameters
#-------------------------------------------------------------#
if [[ $# != 0 ]]
then
  error_msg "$0: incorrect number of input parameters"; return 1
fi
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# start date-time of run
#-------------------------------------------------------------#
typeset sdate=$(print $ddtg | cut -c1-8)
typeset stime=$(print $ddtg | cut -c9-10)0000
#-------------------------------------------------------------#


#-------------------------------------------------------------#
# output interval and number of outputs
#-------------------------------------------------------------#
```

```
typeset dout=$(shvar doutp $prjDir/setup_ww3_multi) \
  || { error_msg "$0:$LINENO: get ww3 doutp failed"; return 1; }
typeset nout=$(( 1 + fcst_length * 3600 / dout ))
#-------------------------------------------------------------------#


#-------------------------------------------------------------------#
# generate namelist
#-------------------------------------------------------------------#
cat << end_namelist
$ ------------------------------------------------------------ $
$ WAVEWATCH III Point output post-processing                    $
$ ------------------------------------------------------------ $
$ First output time (yyyymmdd hhmmss), increment of output (s),
$ and number of output times.
$
  ${sdate} ${stime}  ${dout}  ${nout}
$
$ Points requested ------------------------------------------- $
$ Define points for which output is to be generated.
$
  1
$ mandatory end of list
 -1
$
$ Output type ITYPE [0,1,2,3]
$
  1
$ ------------------------------------------------------------ $
$ ITYPE = 0, inventory of file.
$          No additional input, the above time range is ignored.
$
$ ------------------------------------------------------------ $
$ ITYPE = 1, Spectra.
$         - Sub-type OTYPE :  1 : Print plots.
$                             2 : Table of 1-D spectra
$                             3 : Transfer file.
$                             4 : Spectral partitioning.
$         - Scaling factors for 1-D and 2-D spectra Negative factor
$           disables, output, factor = 0. gives normalized spectrum.
$         - Unit number for transfer file, also used in table file
$           name.
$         - Flag for unformatted transfer file.
$
    1   0.  0.  33  F
$
$ The transfer file contains records with the following contents.
$
$ - File ID in quotes, number of frequencies, directions and points.
$   grid name in quotes (for unformatted file C*21,3I,C*30).
$ - Bin frequenies in Hz for all bins.
```

```
$ - Bin directions in radians for all bins (Oceanographic conv.).
$                                        -+
$ - Time in yyyymmdd hhmmss format                        | loop
$                                    -+          |
$ - Point name (C*10), lat, lon, d, U10 and    |  loop     | over
$   direction, current speed and direction     |   over    |
$ - E(f,theta)                               |  points   | times
$                                    -+          -+
$
$ The formatted file is readable usign free format throughout.
$ This datat set can be used as input for the bulletin generator
$ w3split.
$
$ ------------------------------------------------------------ $
$ ITYPE = 2, Tables of (mean) parameter
$         - Sub-type OTYPE :  1 : Depth, current, wind
$                             2 : Mean wave pars.
$                             3 : Nondimensional pars. (U*)
$                             4 : Nondimensional pars. (U10)
$                             5 : 'Validation table'
$                             6 : WMO standard output
$         - Unit number for file, also used in file name.
$
$ 2  33
$
$ If output for one point is requested, a time series table is made,
$ otherwise the file contains a separate tables for each output time.
$
$ ------------------------------------------------------------ $
$ ITYPE = 3, Source terms
$         - Sub-type OTYPE :  1 : Print plots.
$                             2 : Table of 1-D S(f).
$                             3 : Table of 1-D inverse time scales
$                                 (1/T = S/F).
$                             4 : Transfer file
$         - Scaling factors for 1-D and 2-D source terms. Negative
$           factor disables print plots, factor = 0. gives normalized
$           print plots.
$         - Unit number for transfer file, also used in table file
$           name.
$         - Flags for spectrum, input, interactions, dissipation,
$           bottom and total source term.
$         - scale ISCALE for OTYPE=2,3
$                             0 : Dimensional.
$                             1 : Nondimensional in terms of U10
$                             2 : Nondimensional in terms of U*
$                            3-5: like 0-2 with f normalized with fp.
$         - Flag for unformatted transfer file.
$
$ 1  0. 0. 50   T T T T T T  0  F
$
```

```
$ The transfer file contains records with the following contents.
$
$ - File ID in quotes, nubmer of frequencies, directions and points,
$   flags for spectrum and source terms (C*21, 3I, 6L)
$ - Bin frequenies in Hz for all bins.
$ - Bin directions in radians for all bins (Oceanographic conv.).
$                                          -+
$ - Time in yyyymmdd hhmmss format                        | loop
$                                    -+          |
$ - Point name (C*10), depth, wind speed and   |  loop     | over
$   direction, current speed and direction     |   over    |
$ - E(f,theta) if requested                    |  points   | times
$ - Sin(f,theta) if requested                  |           |
$ - Snl(f,theta) if requested                  |           |
$ - Sds(f,theta) if requested                  |           |
$ - Sbt(f,theta) if requested                  |           |
$ - Stot(f,theta) if requested                 |           |
$                                    -+          -+
$
$ ------------------------------------------------------------ $
$ End of input file                                            $
$------------------------------------------------------------- $
end_namelist
#-----------------------------------------------------------------#


}
#-----------------------------------------------------------------#
# end of function
#-----------------------------------------------------------------#
```

Appendix C: COAMPS Atmospheric Grid Namelist File (gridnl.atmos)

&gridnl

nnest		integer	Number of nested grids	= 3,
kka	integer	Number of atm vertical levels (layers)		= 60,
m	integer	Number of *x* grid positions		= 221, 457,484
n	integer	Number of *y* grid positions		= 137, 283, 475,
nproj	integer	Grid projection number		= 2,
alnnt	real	Standard longitude of grid		=16.547,
phnt1		real	1^{st} standard latitude of grid	=60.0,
phnt2		real	2^{nd} standard latitude of grid	= 30,
rlat	real	Grid reference latitude		=43.456,
rlon	real	Grid reference longitude		=14.264,
iref	integer	i-coordinate of reference point=111,		
jref	integer	j-coordinate of reference point= 69,		
ii	integer	Coarser mesh *x* grid point		= 111,40,136,
jj	integer	Coarser mesh *y* grid point		= 69,13,75,
delx	real	Grid spacing in *x*-direction		= 27000.0,
dely	real	Grid spacing in *y*-direction		= 27000.0,
npgrid		integer	Parent grid number	= 1,1,2,3
lnmove		logical	(true) Moving nest indicator	= f, f, f, f,

Appendix D: NCOM Grid Namelist File (gridnl.ocean)

```
 &gridnl

  alnnt  real              Standard longitude of grid        =241.441,
  delx   real              Grid spacing in x-direction       = 6000.0,
  dely   real              Grid spacing in y-direction       = 6000.0,
  ii               integer Coarser mesh x grid point         =1,50
  iref             integer i-coordinate of reference point   =1,
  jj               integer Coarser mesh y grid point         =1,60,
  jref             integer j-coordinate of reference point   =1,
  kkom             integer Total number of ocean levels      = 37,
  kkosm            integer Total number of sigma levels      =28
  m                integer Number of x grid positions        =262,
  n                integer Number of y grid positions        =886,
  nnest            integer Number of nested grids  = 1,
  npgrid           integer Parent grid number                = 1,1,
  nproj            integer Grid projection number  = 1,
  phnt1            real    1st standard latitude of grid =60,
  phnt2            real    2nd standard latitude of grid = 30,
  rlat   real              Grid reference latitude   =25.00,
  rlon   real              Grid reference longitude          =235.00,

/
```

Appendix E: SWAN Grid Namelist File (gridnl.wave)

&gridnl

alnnt	real	Standard longitude of grid	=241.441,
delx	real	Grid spacing in *x*-direction	= 6000.0,
dely	real	Grid spacing in *y*-direction	= 6000.0,
ii	integer	Coarser mesh *x* grid point	=1,50,
iref	integer	i-coordinate of reference point	= 1,
jj	integer	Coarser mesh y grid point	=1.60,
jref	integer	j-coordinate of reference point	=1,
m	integer	Number of x grid positions	=262,
n	integer	Number of y grid positions	=886,
nnest	integer	Number of nested grids	= 1
npgrid	integer	Parent grid number	= 1,1,
nproj	integer	Grid projection number	= 1,
phnt1	real	1st standard latitude of grid	=60,
phnt2	real	2nd standard latitude of grid	=30,
rlat	real	Grid reference latitude	=25.00,
rlon	real	Grid reference longitude	=235.00

/

[i] COAMPS® is a registered trademark of the Naval Research Laboratory.